REPORT SSD-TR-89-48

(4)

DTIC FILE COPY

AD-A211 670

Batch LCAP2–Linear Control Analysis, Version 2.0: User's Manual

E. A. LEE
Vehicle and Control Systems Division
Engineering Group
The Aerospace Corporation
El Segundo, CA 90245–4691

January 1989

Final Report

Prepared for

SPACE SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
Los Angeles Air Force Base
P.O. Box 92960
Los Angeles, CA 90009-2960

DTIC
S ELECTE D
AUG 28 1989
E

APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED

89 8 22 010

This report was submitted by The Aerospace Corporation, El Segundo, CA 90245-4691, under Contract No. F04701-88-C-0089 with the Space Division, P.O. Box 92960, Los Angeles, CA 90009-2960. It was reviewed and approved for The Aerospace Corporation by R. J. Skrinska, Director, Control Analysis Department. The project officer is Lt. Col. R. W. Proctor, CLDE.

This report has been reviewed by the Public Affairs Office (PAS) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication. Publication of this report does not constitute Air Force approval of the report's findings or conclusions. It is published only for the exchange and stimulation of ideas.

Robert W. Proctor, Lt. Col., USAF
Chief, Titan IV Engineering Division

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

3. DISTRIBUTION / AVAILABILITY OF REPORT
Approved for public release; distribution unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
TR-0089(9975)-1

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
The Aerospace Corporation

6b. OFFICE SYMBOL *(If applicable)*

7a. NAME OF MONITORING ORGANIZATION
Space Division
Air Force Systems Command

6c. ADDRESS *(City, State, and ZIP Code)*
2350 E. El Segundo Blvd.
El Segundo, CA 90245-4691

7b. ADDRESS *(City, State, and ZIP Code)*
Los Angeles Air Force Base
P. O. Box 92960
Los Angeles, CA 90009-2960

8a. NAME OF FUNDING / SPONSORING ORGANIZATION
Space Division

8b. OFFICE SYMBOL *(If applicable)*

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
F04701-88-C-0089

8c. ADDRESS *(City, State, and ZIP Code)*

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*
Batch LCAP2 - Linear Control Analysis Program, Version 2.0: User's Manual

12. PERSONAL AUTHOR(S)
Eugene A. Lee

13a. TYPE OF REPORT

13b. TIME COVERED
FROM ________ TO ______

14. DATE OF REPORT *(Year, Month, Day)*
31 January 1989

15. PAGE COUNT
463

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Control analysis program, Linear systems, Stability analysis, Launch vehicle stability verification,
(Continued)

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Batch LCAP2 (Linear Controls Analysis Program) is a FORTRAN program which provides the controls analyst with a set of easy to use FORTRAN subroutines which implements classical SISO control analysis techniques such as transfer function evaluation, transfer function algebra, frequency response, root locus, inverse time response, and sampled-data transforms. It can handle continuous systems and continuous-discrete multirate systems with the use of s, z, and w transforms. Recent additions to LCAP2 include (1) the capability for connecting transfer function blocks for both continuous and continuous-discrete multirate systems and (2) a precompiler to aid the user in writing FORTRAN code. The methodology used for connecting transfer function blocks takes into consideration the dimensionality problem associated with the use of nondynamic blocks or algebraic variables used in modeling the connection of dynamic

(Continued)

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL

22b. TELEPHONE *(Include Area Code)*

22c. OFFICE SYMBOL

83 APR edition may be used until exhausted.
All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

18. SUBJECT TERMS (Continued)

Frequency response, Root locus, Time response, Laplace transform, Sampled-data, z transform, w transform, Multirate sampled-data, Block diagram connection, Kalman-Bertram state space method

19. ABSTRACT (Continued)

blocks. In the formulation of the system matrix used in LCAP2, the dimension of the system matrix is not increased by the use of nondynamic blocks. The automated analysis of continuous-discrete multirate systems modeled by a connection of transfer function blocks uses the Kalman-Bertram state space method.

LCAP2 has been used extensively for over 20 years in support of various launch vehicle and satellite control systems stability analysis studies. It is the primary frequency domain analysis program used for stability verification of launch vehicles.

Originally developed to run on the CDC mainframe computer, batch LCAP2 now runs on the CRAY XMP/14 as well.

Accession For
Dist
A-1

QUALITY INSPECTED 2

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Batch LCAP2 (Linear Controls Analysis Program) is a FORTRAN program which provides the controls analyst with a set of easy to use FORTRAN subroutines which implements classical SISO control analysis techniques such as transfer function evaluation, transfer function algebra, frequency response, root locus, inverse time response, and sampled-data transforms. It can handle continuous systems and continuous-discrete multirate systems with the use of s, z, and w transforms. Recent additions to LCAP2 include (1) the capability for connecting transfer function blocks for both continuous and continuous-discrete multirate systems and (2) a precompiler to aid the user in writing FORTRAN code. The methodology used for connecting transfer function blocks takes into consideration the dimensionality problem associated with the use of nondynamic blocks or algebraic variables used in modeling the connection of dynamic blocks. In the formulation of the system matrix used in LCAP2, the dimension of the system matrix is not increased by the use of nondynamic blocks. The automated analysis of continuous-discrete multirate systems modeled by a connection of transfer function blocks uses the Kalman-Bertram state space method.

1.1 Background

The original version of this program, LCAP [1], first developed in the late 1960's in support of the analysis of the Titan IIIC launch vehicle digital autopilot, provided the controls analyst with a tool for analyzing complex continuous and continuous-discrete systems using classical transform methods. The key features of this program were the ability to (1) compute z and w transforms and to express them as rational functions, (2) perform transfer function algebra, (3) evaluate transfer functions using Cramer's method, and (4) specify control analysis functions such as frequency response, root locus, or time response with simple commands.

Use of this program consisted of preparing an input card deck in which only numbers could be assigned to input variables. While capable of analyzing the job at hand, it soon became apparent that the program would be more flexible and useful if the user could enter and manipulate algebraic expressions as well as numerical values for the input variables. This flexibility could only be provided if either (1) a higher level user's language was developed or (2) the user was required to write his/her own FORTRAN program. The latter method was chosen and program LCAP2 [2] was developed in the late 1970's. All the commands in LCAP were implemented so that they could be easily invoked in LCAP2 by simple FORTRAN CALL statements. Program LCAP2 is thus a user written FORTRAN program utilizing the LCAP2 Subroutine Library. All the FORTRAN language facilities such as DO LOOPs, GO TOs, user-written SUBROUTINEs and FUNCTIONs, etc., can be utilized to develop a customized control analysis program for specific projects.

As the complexity and degree of the systems being analyzed by LCAP2 grew, two important issues had to be addressed. They were (1) computational accuracy and (2) ease in setting up an analysis. Up until 1987, LCAP2 could only handle transfer functions which were less that 50-th order. For continuous systems, many different control systems were analyzed without computational problems. The setup effort for some of the higher order continuous systems were more complex but were manageable. For discrete systems, computational accuracy has always been a factor for systems greater than 10~20-th degree if an analysis were performed in the z-plane using transfer function algebra.[1] The usual practice to avoid this type of inaccuracy is to analyze discrete systems in the w-plane instead of the z-plane. This method of analysis of discrete systems has been successfully applied to many different single rate systems and to several multirate systems.

For more complex multirate systems in which (1) the ratio of the sampling rates are not small integers and/or (2) when there are too many feedback loops or coupled states, classical z or w transform analysis using transfer function algebra becomes very difficult or impractical to apply. Computational errors associated with transfer function algebra becomes more of a factor as the number of operations required to set up an analysis is increased.

Effort was begun in 1985 to develop an alternate method for evaluating SISO transfer functions which (1) does not employ transfer function algebra, (2) uses state space representation and algorithms for greater accuracy, (3) automates the procedure so that the user needs only to specify the connection of transfer functions, and (4) uses the Kalman-Bertram state space method [3] for analysis of continuous-discrete multirate systems.

After completion of an automated method for transfer function analysis for both continuous and continuous-discrete multirate systems, additional flexibility in using LCAP2 was achieved by the development of the PRECMP preprocessor which can be used to help the user write FORTRAN code.

1.2 Versions

Up until 1987, batch LCAP2 was hosted only on the CDC mainframe 176 and 860 computers. This program is now available on the CRAY XMP-14. In fact two versions of batch LCAP2 are now available on the CRAY. One is identical to the CDC version and the other is a large model version which enables systems twice as large to be analyzed. Differences between these versions of batch LCAP2 and interactive LCAP2 are given below.

1.2.1 CDC Batch LCAP2

There are two versions of batch LCAP2 on the CDC machines, one on the upper CYBER 176 and one on the lower CYBER 860. The maximum transfer function and polynomial order for both

[1] Since roots of z-plane transfer functions are usually all within the unit circle, coefficients of a z-plane transfer function (which are related to the products of the roots) cannot be adequately represented in a computer program except for small order systems. Any addition or subtraction of z-plane coefficients (except for small order systems) carried out by transfer function algebra will result in loss of accuracy.

of these versions is 49-th degree. The number of polynomials and s plane, z plane, and w plane transfer functions which the user can define and use is 999 each.

For transfer function evaluation from a set of Laplace transformed differential equations using Cramer's method, the order of the matrix (with polynomial elements up to 4-th order) is limited to 30x30 with the constraint that the determinant must not be greater than 49-th order.

For automated transfer function analysis using the transfer function connection capability, the limits on the number of blocks that can be connected are: (1) for continuous systems, 30 continuous blocks and (2) for continuous-discrete multirate systems, 20 continuous blocks, 20 discrete blocks, and 5 sample-hold blocks. In addition to the constraint on the number of blocks that can be connected, the characteristic polynomial to be computed must not be greater than 49-th order. For continuous-discrete multirate systems there are additional constraints. The continuous blocks and the discrete blocks, each individually, must not generate more than 30 dynamics states. Also the number of time states used to define the transition matrix must not exceed 50.

1.2.2 CRAY Batch LCAP2

There is a regular and a large model version of batch LCAP2 on the CRAY. The regular version is identical to the CDC version in terms of model size limitations.

For the large model version of LCAP2 the maximum transfer function and polynomial order is 100-th degree. The number of polynomial and transfer functions which the user can use is unchanged.

For transfer function evaluation from a set of Laplace transformed differential equations using Cramer's method, the size of the matrix (with polynomial elements up to 4-th order) for the large model version of LCAP2 is limited to 50x50 with the constraint that the determinant must not be greater than 100-th order.

For automated transfer function analysis using the transfer function connection capability, the limit on the number of blocks that can be connected for the large model version of LCAP2 are: (1) for continuous systems, 75 continuous blocks and (2) for continuous-discrete multirate systems, 50 continuous blocks, 50 discrete blocks, and 10 sample-hold blocks. In addition to the constraint on the number of blocks that can be connected, the characteristic polynomial to be computed must not be greater than 100-th order. For continuous-discrete multirate systems there are additional constraints. The continuous blocks and the discrete blocks, each individually, must not generate more than 75 dynamics states. The limit on number of time states used to define the transition matrix is the same as the regular version of LCAP2.

1.2.3 Interactive LCAP2

Interactive LCAP2 runs on the CDC 860 computer under the INTERCOM 5.0 system and the NOS 1 operating system. The available memory allocated by the operating system for interactive use limits the size of problems which LCAP2 can analyze interactively. The maximum transfer function order and the limit on the number of states for automated transfer function analysis are the same as those for CDC batch LCAP2.

Interactive LCAP2 has been revised extensively since the user's manual [2] was issued. Major changes have been (1) the addition and use of default values to simplify the number of keystrokes, (2) improved capability to save and load data files, and (3) the addition of transfer function connection commands for automated transfer function analysis. The modeling of transfer function connections is much simpler to do in interactive LCAP2. In fact, the fastest method for modeling a complex model is to do the initial connection interactively, save the connection data to a file, and then load this file into a batch job. The user's manual for interactive LCAP2 will be revised in the near future.

1.3 Changes Affecting Users of Previous Versions of LCAP2

In developing this version of batch LCAP2, emphasis was placed on upward compatibility with the previous versions of the program. The names and usage of the LCAP2 commands and LCAP2 data parameters were retained as much as possible. However, several fundamental changes were made which will affect users of previous versions of batch LCAP2. They are described in Appendix C and should be read by all users of previous versions of LCAP2. The changes include (1) the use of CHARACTER variables for labeling plots (previously REAL variables with Hollerith characters were used), (2) changing the use of common block HEADDB to common blocks HEAD, DBASE, POLYCM, and ROOTCM, (3) renaming of LCAP2 array parameters POLY and ROOT to POLYP and ROOTP, respectively, (4) renaming some of the LCAP2 parameters, (5) changing the commands STORE and RESTORE to SAVE and LOAD, respectively, and (6) changing the setup for creating the main FORTRAN program using UPDATE COMDECKs.

The previous version of this program was written in FORTRAN 66 (CDC FTN4). This version is written in FORTRAN 77 (CDC FTN5 and CRAY CFT). To use batch jobs developed on previous versions of LCAP2, the user must make the appropriate code conversion from FORTRAN 66 to FORTRAN 77.

1.4 Organization of User's Manual

After this introductory chapter an overview of the control system analysis features in LCAP2 is given in Chapter 2. A batch program with all these features requires a fair amount of writing to adequately describe its capability. The typical user, however, will seldom read a user's manual from cover to cover, but instead will go directly to a subject of interest, particularly those with examples close to the problem at hand. With this in mind, many examples are included in this manual. Some are complete examples showing actual output from a batch job. Others are code fragments illustrating the utility of the various commands, parameters, and directives. At times the examples and explanations may appear repetitive, but if it can save the user some time it serves its purpose.

The chapters are presented in an order which does not present too much detail initially so as not to detract the user from the utility and description of the batch job structure. Chapters 3 and 4 provide details on the batch job structure and the programming aids available for creating a batch LCAP2 job. If a user is not interested in this level of detail and wants only to run a job as quickly as possible, the best thing to do is to go directly to the examples in Chapters 8 and 9. Most of those examples can be reproduced by the user from files listed in those two chapters.

Chapter 6 describes the data structures used by LCAP2. These include commands, parameters, polynomials, and transfer functions which were defined in the original version of LCAP2. New data structures, namely continuous, discrete, and sample-hold connection blocks, are defined for use in connecting transfer functions as part of the automated transfer function analysis capability.

Chapter 7 describes the new procedure for automated analysis of systems modeled as a connection of transfer function blocks. Examples are presented in Chapters 8 and 9. Basic LCAP2 commands are demonstrated in Chapter 8 while advanced examples, including IEEE CACSD benchmark problems, are presented in Chapter 9. Appendices on various subjects of interest are included at the end of the manual.

Chapter 2

Overview of Control Analysis Capabilities

LCAP2 is a control analysis program which enables the user to easily perform operations on transfer functions and polynomials on a functional level. The transfer functions are defined as $\mathbf{SPTF}_i$, $\mathbf{ZPTF}_i$, and $\mathbf{WPTF}_i$. and the polynomials are defined as $\mathbf{POLY}_i$, where i ranges from 0 through 999. LCAP2 commands are defined so that operations on transfer functions can be implemented with only references to the indices of the transfer functions or polynomials. For example, to add $\mathbf{SPTF}_2$ to $\mathbf{SPTF}_3$ and store the sum into $\mathbf{SPTF}_1$, i.e,

$$\mathbf{SPTF}_1 = \mathbf{SPTF}_2 + \mathbf{SPTF}_3$$

the FORTRAN statement to implement this operation is: `CALL SPADD(1,2,3)`

The arguments for an LCAP2 commands are not always included in the argument list of the FORTRAN subroutine which implements the command. For example, to compute the z transform of $\mathbf{SPTF}_3$ with a sampling period of 0.2 second and to store the results into $\mathbf{ZPTF}_5$, the FORTRAN statements can be written as:

```
SAMPT=.2
CALL SZXFM(5,3)
```

The sampling period is not included as an argument in subroutine SZXFM, but is specified by the value of SAMPT which is in a common block. The rationale for not including all required arguments of a command in the subroutine argument list is given below:

- Argument list is kept short for simplicity.
- Long argument lists increase the chances for FORTRAN errors.
- Use of parameters which are in a common block allows the same parameter to be used by other commands without requiring the user to reenter the same value again.
- If the same command is used more than once, parameters in common blocks do not have to be reentered unless their values have been changed, i.e., plot parameters and plot options do not have to be repeated for each plot unless there are changes.

In this program these parameters will be called LCAP2 parameters.

Having defined the use of **SPTF**$_i$, **ZPTF**$_i$, **WPTF**$_i$, and **POLY**$_i$ and also the use of LCAP2 parameters as additional arguments for LCAP2 commands, an overview of the major control analysis features of LCAP2 will be presented in the following sections. Sections 2.1 through 2.6 describe the basic LCAP2 commands that the user can use for analyzing control systems. For simple systems, these basic LCAP2 commands would be sufficient to perform an analysis based on block diagram reduction using transfer function algebra. However, for more complex systems, transfer function algebra is too difficult to apply or else leads to poor numerical results. There are two methods in LCAP2 for handling these cases. The first one is a method which was first used in the 1960's for analyzing control systems of launch vehicles. This method utilizes Cramer's method for evaluating a transfer function from a set of Laplace transformed differential equations. This technique is described in Section 2.7. The other method for analyzing complex systems is the transfer function block connection method in which the program uses state space methods to automatically compute eigenvalues, transfer functions, root loci, frequency, and time response. This technique is described in Section 2.8. The last section of this chapter describes the restart capability of the program.

2.1 Commands for Transfer Function Algebra

A set of LCAP2 commands is defined so that the user can easily perform transfer function algebra with a simple FORTRAN CALL statement such as the SPADD command described previously. In using these commands, the user only has to reference the indices of the **SPTF**$_i$, **ZPTF**$_i$, **WPTF**$_i$, or **POLY**$_i$ involved. The user does not have to be concerned if the transfer function or polynomial is in coefficient or root (factored) form. The data base for **SPTF**$_i$, **ZPTF**$_i$, **WPTF**$_i$, and **POLY**$_i$ allows both forms to be saved. In fact, each transfer function and polynomial will have an internal flag to indicate if the roots are available so that LCAP2 commands can be implemented with better accuracy. For example, if two polynomial or transfer functions are to be multiplied, and their roots are available, the product is computed by collecting the roots rather than by multiplying the coefficients. Along this line of implementation, if two transfer functions are to be added or subtracted, the program first checks to see if the roots of the transfer functions are available. If they are available, any common roots between the denominators are factored out before the sum is computed. Other steps taken to improve the numerical results include automatic cancellation of common numerator and denominator roots where appropriate and automatic deletion of negligible high order coefficients in an add or subtract operation.

The commands for transfer function algebra are given in Table 2.1. Included in this table are commands for loading, printing, and deleting polynomials and transfer functions.

Table 2.1: Commands for Transfer Function Algebra

PADD	Polynomial add	
PEQU	Polynomial equal	
PSUB	Polynomial subtract	
PMPY	Polynomial multiply	
PLDC	Polynomial load, coefficient form	
PLDR	Polynomial load, root form	
POLY	Print out polynomial	
PDEL	Delete polynomial	
SPADD	S plane transfer function add	*
SPEQU	S plane equal	*
SPSUB	S plane transfer function subtract	*
SPMPY	S plane transfer function multiply	*
SPDIV	S plane transfer function divide	*
SPCLSLP	S plane closed loop transfer function	*
SELCR	Eliminate common roots of an s plane transfer function	*
SNORM	S plane transfer function normalization	*
SPLDC	S plane load, coefficient form	*
SPLDR	S plane load, root form	*
SPTF	Print out s plane transfer function	*
SPDEL	Delete s plane transfer function	*
CPYPS	Copy polynomials into an s plane transfer function	*
CPYSP	Copy s plane transfer function into 2 polynomials	*

* - Corresponding z and w commands are available by substituting Z or W for the letter S.

2.2 Frequency Response

Frequency responses of transfer functions can be computed. Bode, Nichols, and Nyquist plots of the responses can be generated. Both low resolution printer plots and high resolution hardcopy plots are available. For the high resolution plots, the points are not connected when there are discontinuities, such as when the phase in a Nichols plot crosses from one side to the other. Two methods are available for choosing the frequency points used in evaluating the frequency response, an automatic and a nonautomatic mode. In the automatic mode the user specifies up to 20 preselected points. The program will then use these points plus its own dynamically generated points to yield a smooth plot so that all resonances of a response will be plotted. In the nonautomatic mode the user can specify up to five different frequency segments in which each segment contains frequencies which are linearly spaced. With both modes, the program will detect gain and phase crossovers and iterate to compute the exact crossover frequency.

Frequency responses can be computed for four different types of transfer functions.

- Commands SFREQ, ZFREQ, and WFREQ are used to compute, respectively, the response of **SPTF**$_i$, **ZPTF**$_i$, and **WPTF**$_i$ transfer functions which are all rational functions.

- Command B1FREQ is used to compute the response of an s plane transfer function defined by a connection s plane transfer function blocks. This command evaluates the response directly from a state space representation of the system without solving for the poles and zeros of the transfer function. Command B2FREQ is used to compute the response of a z plane transfer function defined by a connection of s and z plane transfer functions and sample-hold blocks. Like the B1FREQ command, the response is computed without having to solve for the poles and zeros of the transfer function.

- Command FREQS, FREQZ, and FREQW are used to compute the response of a general function which the user specifies in a user-defined function. For example, these functions can be (1) a nonrational function such as the closed loop transfer function of a system with dead time or (2) a transfer function which is not explicitly represented as a ratio of two polynomials, i.e., a ratio of two matrices with polynomial or transfer function elements. These user-defined function frequency response commands are also referred to as generalized frequency response commands.

- Command ZMRFQ is used to numerically evaluate the fast-slow multirate (by frequency decomposition) frequency response of a **ZPTF**$_i$ transfer function. Command WMRFQ is the w plane equivalent of command ZMRFQ.

The frequency response commands are given in Table 2.2.

Table 2.2: Frequency Response Commands

SFREQ	Frequency response of **SPTF**$_i$
ZFREQ	Frequency response of **ZPTF**$_i$
WFREQ	Frequency response of **WPTF**$_i$
B1FREQ	SISO frequency response of a continuous system modeled as a connection of s plane transfer functions
B2FREQ	SISO frequency response of a continuous-discrete multirate system modeled as a connection of s and z plane transfer functions and sample-hold blocks
FREQS	Frequency response of a user supplied s plane function
FREQZ	Frequency response of a user supplied z plane function
FREQW	Frequency response of a user supplied w plane function
ZMRFQ	Frequency response of a fast-slow multirate z transform
WMRFQ	Frequency response of a fast-slow multirate w transform

2.3 Time Response

Time response of (1) **SPTF**$_i$ and **ZPTF**$_i$ transfer functions and (2) continuous and continuous-discrete systems defined by a connection of transfer function blocks can be computed.

For an **SPTF**$_i$ transfer function, the inverse Laplace transform is computed by the partial fraction method to yield an analytical solution. The time response is obtained by evaluating it over a time period of interest. Inputs for this command are (1) type of input, an impulse or step, (2) magnitude, (3) start time, (4) end time, (5) delta time, and (6) plot parameters.

For a **ZPTF**$_i$ transfer function, the response is computed by recursive evaluation of a difference equation. The input for this command is similar to that for the s plane case except that the start time must be $t = 0$ and the response is computed at every sampling time.

For continuous and continuous-discrete systems defined by a connection of transfer function blocks, state space methods are used to compute a SISO time response. The inputs for these commands are similar to the command for time response of a **ZPTF**$_i$ transfer function.

Both low resolution printer plots and high resolution hardcopy plots of the response are available.

The time response commands are given in Table 2.3.

Table 2.3: Time Response Commands

STIME	Time response of **SPTF**$_i$ by inverse Laplace transform
ZTIME	Time response of **ZPTF**$_i$ by recursive evaluation of a differenece equation
B1TIME	SISO time response of a continuous system modeled by a connection of s plane transfer functions
B2TIME	SISO time response of a continuous-discrete multirate system modeled as a connection of s and z plane transfer functions and sample-hold blocks

2.4 Root Finding

Roots of a **POLY**$_i$ polynomial or an **SPTF**$_i$, **ZPTF**$_i$, or **WPTF**$_i$ transfer function can be computed. The root finding commands are given in Table 2.4.

Table 2.4: Root Finding Commands

PRTS	Find roots of **POLY**$_i$
SPRTS	Find roots of **SPTF**$_i$
ZPRTS	Find roots of **ZPTF**$_i$
WPRTS	Find roots of **WPTF**$_i$

2.5 Root Locus

Root locus for an $\mathbf{SPTF}_i$, $\mathbf{ZPTF}_i$, and $\mathbf{WPTF}_i$ transfer function can be computed. The transfer function must represent the open loop transfer function of the system being analyzed. The gains used for computing the root locus can be selected from a combination of fixed gains and linearly or logarithmic spaced gains. The roots for each gain is printed out.

Both low resolution printer plots and high resolution hardcopy plots are available. For hardcopy plots, an option is available for tagging each gain.

The root locus commands are given in Table 2.5.

Table 2.5: Root Locus Commands

SLOCI	Root locus of $\mathbf{SPTF}_i$
ZLOCI	Root locus of $\mathbf{ZPTF}_i$
WLOCI	Root locus of $\mathbf{WPTF}_i$

In using the commands in Table 2.5 to compute the root locus of a system, the open loop transfer function must first be computed. In general, root locus plots do not necessarily have to correspond to variations in a loop gain. The loci can be with respect to any variable. To use the commands in Table 2.5, all that is necessary is that an appropriate open loop transfer function be computed. This, however, is not always easy to do, particularly with complex systems in which the gain to be varied is not directly in a forward or feedback loop. For these situations commands B1LOCI and B2LOCI described in Section 2.8 can be used to automatically compute the root loci without first computing an open loop transfer function.

2.6 Transfer Function Transforms

Many different transforms between $\mathbf{SPTF}_i$, $\mathbf{ZPTF}_i$, and $\mathbf{WPTF}_i$ transfer functions are available. These commands generally have names ending with the letters XFM. The first two letters of the command generally describe the transform. For example, the command SZXFM computes the classical s to z plane transform by the partial fraction method. This command includes a time delay and a zero order hold if desired. The analogous command for s to w plane is SWXFM.

To convert between the z and w plane there are two bilinear transform commands, ZWXFM and WZXFM. The bilinear transforms are defined by the following:

$$w = \frac{z-1}{z+1}$$

$$z = \frac{1+w}{1-w}$$

Although the transform from the z or w plane to the s plane is not unique, commands ZSXFM and WSXFM are defined to compute "equivalent" s plane roots to aid the analyst in identifying and correlating z and w plane roots. These two commands do not generate an s plane transfer function; they only print out the equivalent s plane roots. The equivalent s plane roots are computed from the following:

$$s = ln(z)/T$$

$$s = ln(\frac{1+w}{1-w})/T$$

where T is the sampling period.

For multirate analysis there are three types of commands:

1. Commands SZMRX and SWMRX are identical to commands SZXFM and SWXFM, respectively, except that when the zero-order hold is used, the sampling period of the zero-order hold is at the slower rate.

2. Command ZVCNG changes the z variable of a z plane transfer function to z_n^n.

3. Commands ZMRXFM and WMRXFM compute the fast-slow multirate transform by frequency decomposition. The resulting transform is a rational function at the slower rate variable.

The transfer function transform commands are given in Table 2.6.

Table 2.6: Transfer Function Transform Commands

SZXFM	S to z plane transform (classical z transform)
SWXFM	S to w plane transform (classical w transform)
ZWXFM	Z to w plane bilinear transform
WZXFM	W to z plane bilinear transform
ZSXFM	Z to "equivalent" s plane root transform
WSXFM	W to "equivalent" s plane root transform
SZMRX	S to z plane slow-fast multirate transform (ZOH at a slower rate)
SWMRX	S to w plane slow-fast multirate transform (ZOH at a slower rate)
ZMRXFM	Z plane fast-slow multirate transform in rational form
WMRXFM	W plane fast-slow multirate transform in rational form

2.7 Transfer Function Evaluation from a Set of Laplace Transformed Differential Equations

Modeling of control systems by using transfer function algebra for block diagram reduction is straightforward if the configuration is not complex and all the transfer functions are given as ratios of polynomials. In the analysis of large launch vehicles both of these considerations are not present. There are many feedback loops and the vehicle dynamics equations are very complex so that transfer functions between actuators and sensor outputs cannot be determined symbolically. This type of problem can be solved by converting all the transfer functions (compensators, actuators, and sensors) to sets of Laplace transformed equations and then augmenting them with the vehicle dynamics equations to form a system represented by:

$$\mathbf{M}(s)\ \mathbf{x}(s) = \mathbf{B}(s)\ u(s)$$

where, $\mathbf{M}(s) = M4\ s^4 + M3\ s^3 + M2\ s^2 + M1\ s + M0$
$\mathbf{B}(s) = B4\ s^4 + B3\ s^3 + B2\ s^2 + B1\ s + B0$
$\mathbf{x}(s)$ is a state vector
M0, M1, M2, M3, M4 are square matrices
B0, B1, B2, B3, B4 are column vectors
u(s) is a scalar

Then by Cramer's method, the transfer function from u to x_j is given by:

$$\frac{x_j(s)}{u(s)} = \frac{\det \mathbf{M}_j(s)}{\det \mathbf{M}(s)}$$

where $\mathbf{M}_j(s)$ is equal to $\mathbf{M}(s)$ with column j replaced by $\mathbf{B}(s)$. By definition, $\mathbf{M}_0(s) = \mathbf{M}(s)$.

The commands for evaluating the determinant of $\mathbf{M}_j(s)$ are given in Table 2.7. It includes the old version, DETRM, for compatibility with previous versions of LCAP2.

Table 2.7: Commands for Determinant of a Polynomial Matrix

DTERM	Determinant of $\mathbf{M}_j(s)$
DETRM	Determinant of $\mathbf{M}(s)$ (old version)

In setting up an analysis using the above method, include as state variables all output variables which define an open or closed loop transfer function of interest. Transfer functions are then computed by evaluating the appropriate determinants and using the CPYPS command to store them into an **SPTF**$_i$ transfer function. Frequency response, time response, or root locus commands can then be used to analyze the system.

The above method of analysis can only be used for continuous systems. For analysis of continuous-discrete systems, such as launch vehicles, this method of analysis can be used only to compute s plane transfer functions of the plant which are usually very complex due to flexible bending dynamics. W transforms[1] of the plant transfer functions and other s plane transfer functions (i.e., sensor and actuator) are then computed. Similarly, w transforms of the digital compensators are computed. For most launch vehicles, once the plant dynamics blocks between the actuators and the sensors are computed, the configuration of system in the w plane is not that complex so that transfer function algebra can be used to compute open and closed loop transfer functions for assessing stability and performance of the system.

[1] W plane instead of z plane used for computational reasons.

2.8 Automated Analysis Using Transfer Function Connection Blocks

The automated analysis method using transfer function connection blocks can be used for analysis of both continuous and continuous-discrete multirate systems. The user does not have to perform block diagram reduction using transfer function algebra or write sets of Laplace transformed differential equations for the entire system as described in Section 2.7. Continuous, discrete, and sample-hold connection blocks are defined and used together with the **SPTF**$_i$ and **ZPTF**$_i$ transfer functions so that the user can efficiently set up and change the configuration of a system being analyzed. All that is required of the user for modeling a system is to (1) enter the number of connection blocks, (2) assign a descriptive label to each connection block, (3) specify an **SPTF**$_i$ or **ZPTF**$_i$ identifier number for each continuous or discrete connection block, (4) specify the inputs to each connection block, and (5) specify the sampling period for each discrete and sample-hold connection block.

The continuous, discrete, and sample-hold connection blocks are defined as **C**$_i$, **D**$_i$, and **S**$_i$, where i ranges from 1 to the number of blocks defined. This overview will not go into the details on how to specify the connection blocks. That will be done in Chapter 7. The LCAP2 commands for implementing this type of analysis will have the prefix B1 or B2. B1 will be used for continuous systems and B2 will be used for continuous-discrete multirate systems. An example of both type of systems will be presented in the following subsections to summarize how the blocks are connected and used.

2.8.1 Continuous Systems

The following continuous system in Figure 2.1 has five different s plane connection blocks, each with a desriptive label.

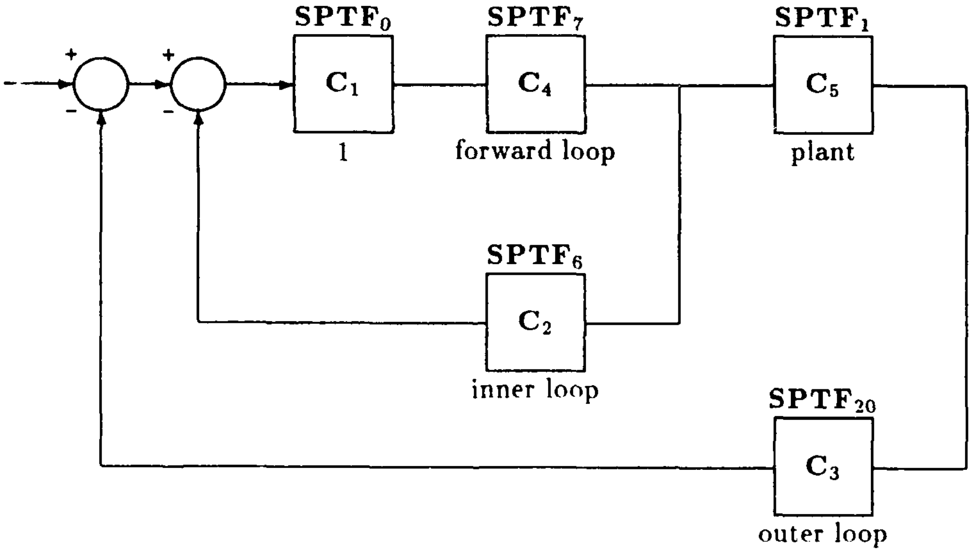


Figure 2.1: Continuous System Example

After the user has specified the connection for each block, the following will be printed by the program:

```
NUMBER OF CONTINUOUS BLOCKS = 5
BLOCK     DESCRIPTION

C1        SUMMING BLOCK USED TO DEFINE -(C2 + C3)
C2        INNER LOOP COMPENSATOR
C3        OUTER LOOP COMPENSATOR
C4        FORWARD LOOP COMPENSATOR
C5        PLANT

BLOCK     TRANSFER FUNCTION           INPUTS FROM BLOCK
C1            SPTF0              -C2  ,-C3
C2            SPTF6               C4
C3            SPTF20              C5
C4            SPTF7               C1
C5            SPTF1               C4
```

Each time a summary of the transfer function connection blocks is printed out, a description of each C_i block is included to aid the user in verifying the correctness of the connections. All the data in the above printout, which completely describes the connection of the system, can be saved to a file so that it can be reused in another job. This transfer function connection file only describes the connection of the C_i blocks. It is totally independent of $\mathbf{SPTF}_i$ data except for the SPTF identifier number associated with each C_i block. This separation between connection of C_i data and $\mathbf{SPTF}_i$ data allows the user to model and analyze a system more efficiently. Revisions to an existing model can be made by substituting different SPTF identifiers for the appropriate C_i blocks.

Unlike the block diagram reduction method described in the previous subsections, opening and closing loops does not entail any effort on the part of the user other than the removal or addition of an input to the appropriate C_i block.

After a system has been modeled, there are five LCAP2 commands that are available for analysis of continuous systems. Command B1EIG will compute the eigenvalues of the system and store the results into a $\mathbf{POLY}_i$ polynomial. Command B1FREQ will compute the s plane frequency response between any two C_i blocks. Command B1TIME will compute the time response between any two C_i blocks. Command B1LOCI will compute the root locus by varying the gain of any C_i block. Command B1TF will compute the transfer function in rational form between two C_i blocks and store the results into an $\mathbf{SPTF}_i$ transfer function.

State space methods are used to implement this automated analysis procedure.

The Block 1 analysis commands are given in Table 2.8.

Table 2.8: Commands for Automated Analysis of Continuous Systems

B1EIG	Compute eigenvalues of a continuous system modeled as a connection of s plane transfer functions
B1FREQ	Compute frequency response of a continuous system modeled as a connection of s plane transfer functions
B1TIME	Compute time response of a continuous system modeled as a connection of s plane transfer functions
B1LOCI	Compute root locus of a continuous system modeled as a connection of s plane of transfer functions
B1TF	Compute transfer function of a continuous system modeled as a connection of s plane transfer functions

2.8.2 Continuous-Discrete Multirate Systems

The continuous-discrete multirate system in Figure 2.2 has two s plane, seven z plane, and one zero order hold block, each with a descriptive label. This example[1] is from the IEEE CACSD Benchmark Problem No. 3, 2-Rate Model, in Ref. 5.

After the user has specified the connection for each block, the following will be printed by the program:

```
NUMBER OF CONTINUOUS BLOCKS = 2
BLOCK     DESCRIPTION

C1        G1 BLOCK
C2        G2 BLOCK

BLOCK     TRANSFER FUNCTION          INPUTS FROM BLOCK
C1            SPTF1              S1

C2            SPTF2              S1
-----------------------------------
NUMBER OF DISCRETE BLOCKS = 6
BLOCK    DESCRIPTION

D1       G3 BLOCK
D2       G4 BLCCK
D3       G5 BLOCK
D4       G6 BLOCK
D5       G7 BLOCK
D6       SUMMER BLOCK
```

[1] The unity D_7 block is not in the benchmark problem. It is included here to define the sampled output of block C_1 at the slower sampling rate. This output state is used for defining a close loop transfer function.

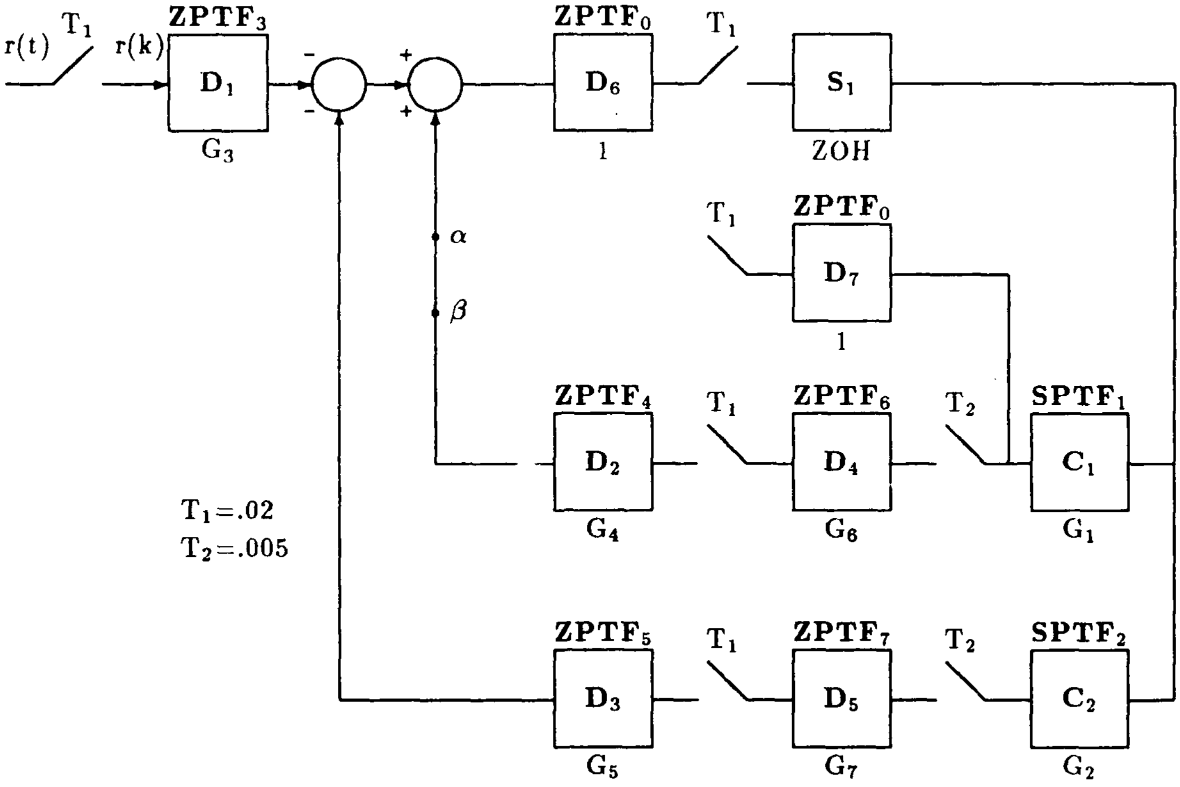


Figure 2.2: Continuous-Discrete Multirate System Example

```
BLOCK    TRANSFER FUNCTION  SAMPLING PERIOD    INPUTS FROM BLOCKS
D1            ZPTF3          .20000E-01      HAS NO INPUT
D2            ZPTF4          .20000E-01      D4
D3            ZPTF5          .20000E-01      D5
D4            ZPTF6          .50000E-02      C1
D5            ZPTF7          .50000E-02      C2
D6            ZPTF10         .20000E-01      -D1   , D2   ,-D3
-----------------------------------
NUMBER OF SAMPLE-HOLD BLOCKS = 1
BLOCK    DESCRIPTION

S1       ZOH BLOCK

BLOCK    SAMPLING PERIOD           INPUTS FROM BLOCKS

S1          .20000E-01           D6
```

Each time a summary of the transfer function connection blocks is printed out, a description of each $\mathbf{C}_i$, $\mathbf{D}_i$, and $\mathbf{S}_i$ block is included to aid the user in verifying the correctness of the connection. Each $\mathbf{D}_i$ and $\mathbf{S}_i$ block includes its sampling period which are all assumed to be synchronized at t=0. All the data in the above printout, which completely describes the connection of the system, can be saved to a file so that it can be reused in another job. This transfer function connection file only describes the connection of the $\mathbf{C}_i$, $\mathbf{D}_i$, and $\mathbf{S}_i$ blocks. It is totally independent of $\mathbf{SPTF}_i$ and $\mathbf{ZPTF}_i$ data except for the SPTF or ZPTF identifier number associated with each $\mathbf{C}_i$ or $\mathbf{D}_i$ block. This separation between connection of $\mathbf{C}_i$ and $\mathbf{D}_i$ data and $\mathbf{SPTF}_i$ and $\mathbf{ZPTF}_i$ data allows the user to model and analyze a system more efficiently. Revisions to an existing model can be made by substituting different SPTF and ZPTF identifiers for the appropriate $\mathbf{C}_i$ or $\mathbf{D}_i$ blocks.

The first part of this automated analysis procedure is to compute the state space representation for the continuous, discrete, and sample-hold parts of the system in terms of the output states of the $\mathbf{C}_i$, $\mathbf{D}_i$ and $\mathbf{S}_i$ blocks. The next part is the use of the Kalman-Bertram state space method to compute the state transition matrix of the system. The sampling period of the computed state transition matrix is equal to the LCM[1] of all the sampling periods and it defined in this manual as the LCM sampling frequency. The last part is computing eigenvalues, transfer functions, root loci, and frequency response from the transition matrix.

There are five LCAP2 commands available for analysis of continuous-discrete multirate systems. Command B2EIG will compute the eigenvalues of the system and store the results into a $\mathbf{POLY}_i$ polynomial. Command B2FREQ will compute the z plane frequency response between two $\mathbf{D}_i$ blocks. Command B2TIME will compute the time response between two $\mathbf{D}_i$ blocks. Command B2LOCI will compute the root locus by varying the gain of any $\mathbf{C}_i$ or $\mathbf{D}_i$ block. Command B2TF will compute the transfer function between two $\mathbf{D}_i$ blocks. For commands B2FREQ, B2TIME, and B2TF the following constraints are imposed:

- A transfer function can only be defined at the LCM sampling period.
- The input and output $\mathbf{D}_i$ blocks which define a transfer function must be at the LCM sampling period.
- Time response can only be evaluated at the LCM sampling period.

These constraints on commands B2FREQ and B2TF means that the frequency response method for evaluating stability of single rate systems cannot always be applied to multirate systems. Frequency response of an open loop can only be computed if the sampling period of the loop is equal to the LCM sampling period. For a loop which is sampled at a rate faster than the LCM sampling rate, a time invariant transfer function cannot be computed. Thus, gain and phase margins can only be computed for those loops which are at the LCM sampling rate. This, however, does not mean that stability of a faster loop cannot be assessed since a root locus analysis can always be performed at any sampling rate.

The Block 2 analysis commands for continuous systems are given in Table 2.9.

2.9 Restart Capability

The restart capability consists of LCAP2 commands for saving and loading two different types of data. The command SAVE will save (1) all $\mathbf{SPTF}_i$, $\mathbf{ZPTF}_i$, $\mathbf{WPTF}_i$, and $\mathbf{POLY}_i$ data, (2) all M0, M1, M2, M3, and M4 matrices and (3) all B0, B1, B3, B4 vectors that are defined at the time

[1] Least-common-multiple

Table 2.9: Commands for Automated Analysis of Continuous-Discrete Multirate Systems

B2EIG	Compute eigenvalues of a continuous-discrete multirate system modeled as a connection of s and z plane transfer functions and sample-hold blocks
B2FREQ	Compute frequency response of a continuous-discrete multirate system modeled as a connection of s and z plane transfer functions and sample-hold blocks
B2TIME	Compute time response of a continuous-discrete multirate system modeled as a connection of s and z plane transfer functions and sample-hold blocks
B2LOCI	Compute root locus of a continuous-discrete multirate system modeled as a connection of s and z plane transfer functions and sample-hold blocks
B2TF	Compute transfer function of a continuous-discrete multirate system modeled as a connection of s and z plane transfer functions and sample-hold blocks

this command is invoked.[1] This data is saved to a local file which can be CATALOGed (CDC) or SAVEed (CRAY) upon exit from a batch job. The data is saved as a sequential unformatted file. Data saved by the command SAVE can be reloaded by the command LOAD.

The commands B1SAVE and B2SAVE will save the transfer function connection data described in Section 2.8. Unlike the SAVE and LOAD commands, the data is saved as an *ASCII file* using a format which is compatible with the PRECMP precompiler.

The purpose of using this format is:

- The file is short.
- The file can be read and edited by an editor.
- Use of PRECMP directives allows more versatile use of the data. This will be described in a following paragraph.

Like the SAVE command, the files created by the B1SAVE and B2SAVE commands are saved to local files which must be CATALOGed (CDC) or SAVEed (CRAY) upon exit from a batch job.

In a restart, the normal method for loading in data saved with either the B1SAVE or B2SAVE commands is to use the B1LOAD or B2LOAD commands. However, since this file was created with a format which is compatible with PRECMP, the data itself can be used to create the FORTRAN source code to define the transfer function connection. This capability would be useful in a fast setup of a batch job for analysis of a system with many connection blocks. Since the FORTRAN code or PRECMP code for specifying the connection blocks will take some effort to set up and check out, the user can use interactive LCAP2 instead to specify the transfer function connections. Interactive LCAP2 is much easier to use for specifying the connection of transfer function blocks

[1] If there are transfer function and polynomial data that are not to be saved, delete them first with the PDEL, SPDEL, ZPDEL, or WPDEL commands.

since it is prompt driven and includes logic tests for reasonable inputs. Once the connections has been completed and verified on interactive LCAP2, the user can save the data with the B1SAVE or B2SAVE command. The data in this file can then be read by an editor and inserted as source code to a batch LCAP2 job which uses the PRECMP precompiler. The amount of time saved can be significant.

The restart commands are given in Table 2.10. All of these commands include a file name as one of its arguments.

Table 2.10: Restart Commands

SAVE	Save all $\mathbf{SPTF}_i$, $\mathbf{ZPTF}_i$,$\mathbf{WPTF}_i$, $\mathbf{POLY}_i$ data, all M0, M1, M2, M3, M4 matrices, and all B0, B1, B2, B3, B4 vectors currently in use
LOAD	Load data saved by the SAVE command
B1SAVE	Save transfer function connection data for a continuous system
B2SAVE	Save transfer function connection data for a continuous-discrete multirate system
B1LOAD	Reload data saved by the B1SAVE command
B2LOAD	Reload data saved by the B2SAVE command

Chapter 3

Overview of Batch LCAP2 Program Structure

Although LCAP2 commands are defined to be easily used by a control systems analyst, the job structure for a batch job is not necessarily simple. This would be the case for users not familiar with the CDC and CRAY UPDATE programs for maintaining source code libraries. To complicate matters, another level of complexity is added to the job structure if the optional PRECMP preprocessor program for generating FORTRAN code is used. Before introducing the use of the UPDATE and PRECMP programs in the next chapter, an overview of the batch LCAP2 job structure is presented.

A batch job consists of (1) a set of control cards and (2) user code for creating a FORTRAN program. The FORTRAN code to be created is a main program plus any subroutines or functions that the user may want to write. The main program must include many different common block declaration statements, some which will require many lines of code. The user is not expected to enter in all this code each time a main program is to be created since mistakes can be easily made. What is needed is a statement or a directive which will allow a set of FORTRAN statements to be inserted into the body of a FORTRAN program. Unfortunately, this is not a part of the standard ANSI FORTRAN language. Some FORTRAN compilers, like the IBM System/360 and Lahey F77L, have an INCLUDE statement as an extension to their language to allow this type of code insertion to be made. Both the CDC and CRAY FORTRAN languages do not have this INCLUDE statement as part of their extended language. To insert sets of source code into a program on these two machines, the UPDATE program is used. This is a line-oriented text editor for maintaining programs in the form of source code. It creates and modifies source code libraries and produces output that can be used as input to other programs, which for LCAP2 is the FORTRAN compiler. All common blocks used by LCAP2 are saved in the program library so that they can be easily inserted into the user's source code with the appropriate UPDATE directive. This will be described in the next chapter.

Another use of the UPDATE program in batch LCAP2 is that it allows access to LCAP2 source code for making modifications. Typically, the only time that a user would access this code is when a user-defined function is to be used. Several user-defined functions are coded so that the user needs only to insert the appropriate statements in the body of the routine to use it.

The output of the UPDATE program is source code which is to be compiled by the FORTRAN compiler. After this code is compiled, it is linked with various libraries, loaded, and executed.

The overview presented so far has not included the PRECMP precompiler for writing FORTRAN statements. This is an optional program that is to be used before the UPDATE program. When it is used, all the user's code, including UPDATE directives will be inputs to the PRECMP program. Output of the PRECMP program is a file named PCMPILE which will be used as input to the UPDATE program.

A flow chart on how the LCAP2 program is created is given in Figure 3.1.

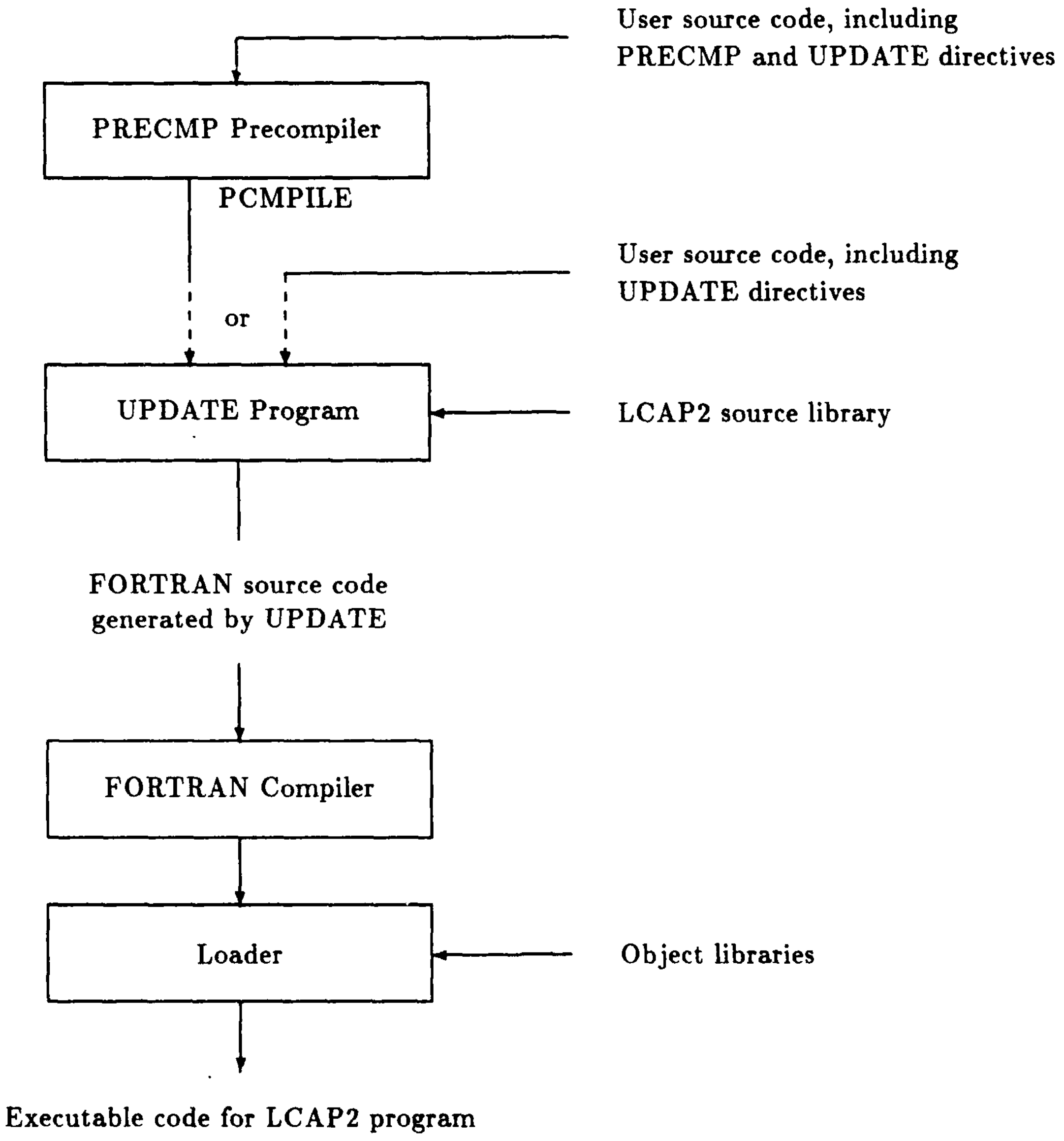


Figure 3.1: Flow Chart of LCAP2 Program Creation

Chapter 4

Program Development Aids

4.1 UPDATE

The UPDATE program is a line-oriented text editor for maintaining programs in the form of source code. This discussion on UPDATE is very limited. Only basic UPDATE operations and directives for coding an UPDATE input file will be discussed. For more detail, see Refs. 6 and 7. An LCAP2 source code library has been created which the user must attach each time a run is to be made. This library includes all the source code for (1) the common blocks used by the LCAP2 subroutine library and the main program which the user is to create and (2) the subroutines in the LCAP2 library. The common blocks are stored in UPDATE COMDECKs which can be copied to any location in the user's source code with the *CALL directive.

When a batch LCAP2 job is executed, the UPDATE program is used to create the source code for the user's main program. In UPDATE terminology, this is called a modification run since an old source code library is to be used. The first record of the input file for UPDATE must be an *IDENT directive with a modification set identifier. This is then followed by one or more modification sets which usually begin with a *INSERT, *BEFORE, or *DELETE directive. For an LCAP2 job, the input file should have the following form:

```
*IDENT identifier
*INSERT START.1
      PROGRAM LCAP2(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
*CALL COMLCAP2
      CALL INITO              "note that the last character is the number 0"
      CALL MINITO             "note that the last character is the number 0"
           ⋮
       user code
           ⋮
      CALL LEXIT
      END
```

where the identifier in record 1 can be any eight character name. The *INSERT directive is used to designate where the next text line image, in this case, the PROGRAM statement, is to be placed. The argument of this directive, START.1, is a located near the beginning of the old source code library before any of the LCAP2 subroutine code. This will ensure that main program will be compiled before any FORTRAN subroutine or function. The PROGRAM statement is the first FORTRAN statement of the main program. The *CALL directive will insert source code from the old program library after the PROGRAM statement. The argument of this *CALL directive is COMLCAP2 which is an UPDATE COMDECK containing many *CALL directives to other LCAP2 COMDECKs. With this one *CALL directive about 89 lines of source code for 20 common blocks needed to run LCAP2 will be inserted. After the *CALL directive there are calls to the intialization routines INIT0 and MINIT0. Subroutine INIT0 is a general initialization routine and MINIT0 is an initialization routine for matrices and vectors used by the DETRM and DTERM commands. These two initialization routines were not combined since it is possible to reduce LCAP2 memory size requirements if the DETRM and DTERM commands are not used. User code follows these initialization routines. The routine LEXIT must be used to terminate a run, or the plot buffer will not be flushed (last plot will be incomplete) when plot jobs are created.

4.2 PRECMP Precompiler

Since some of the LCAP2 commands require many lines of FORTRAN statements to be written by the user, particularly those which involve the input of array data, a precompiler PRECMP was developed to aid the user in writing the required FORTRAN code. For example, to load the following polynomial,

$$As^5 + Bs^4 + A{*}Bs^3 + 45s^2 + 130s + 500$$

into **POLY**$_3$, the following FORTRAN statements are required.

```
POLYP(1)=5                   "degree of polynomial"
POLYP(2)=500.
POLYP(3)=130.
POLYP(4)=45.
POLYP(5)=A*B
POLYP(6)=B
POLYP(7)=A
CALL PLDC(3)
```

The above eight lines of FORTRAN code can also be written by PRECMP using the following two PRECMP directive statements.

```
*POLYP 5 500.  130.  45.  A*B B A
*PLDC 3
```

There are two types of PRECMP directives, the parameter and the command directives. A parameter directive is used only for entering data into a FORTRAN variable. In the example above the *POLYP directive is used for loading in polynomial coefficients into the FORTRAN array POLYP. A command directive is used for invoking an LCAP2 command. In the example above the *PLDC directive is used for loading the data from FORTRAN array POLYP into $\mathbf{POLY}_3$.

A typical batch LCAP2 job setup consists of a set of control cards and an input file. If PRECMP is not used, the input file is read directly by the UPDATE program which will then create a FORTRAN source deck ready for compilation. If PRECMP is used, the user's input file will be read one record at a time by PRECMP. If a record is not a PRECMP directive, i.e., a FORTRAN statement or an UPDATE directive, it will be copied to a file named PCMPILE. If it is a PRECMP directive, the directive will be "cracked" and FORTRAN statements for implementing the directive will be written to file PCMPILE. After all the records have been preprocessed, the file PCMPILE is rewound and used as the input to the UPDATE program.

The typical job setup was described in terms of optionally using PRECMP rather than including it as a standard procedure for preprocessing a user's input file. The rationale for this is based on the slow execution time of the PRECMP program. For small order jobs, the execution time for PRECMP can be as long as the execution time for running the LCAP2 program. This is due to inefficient execution of character operations by the CDC and CRAY computers. Thus, the use of PRECMP will be at the option of the user. Sample job setups are given in Chapter 5 for the CDC and CRAY computers.

4.2.1 PRECMP Directive Format

The general format of the PRECMP directive is shown below in Table 4.1. A directive must begin with the master control character in column one. If a directive statement requires more than one line, two consecutive periods, .. are used as a continuation character to designate that input is to follow on the next line. Any text on the same line after the continuation character, though, is ignored by PRECMP and is considered to be a comment. An optional character ! can be used to terminate a directive statement so that any text following ! is considered to be a comment. Optional characters or parameters for PRECMP directive definition are enclosed by the curly braces, { }. The directive format is similar to the directives used by the CDC and CRAY UPDATE programs. Since the output of PRECMP is used as input to UPDATE, all UPDATE directives and their abbreviations are ignored by PRECMP.

Table 4.1: Format of PRECMP Directive

*keyword p-list {..} {!}	
*	Master control character that distinguishes a directive from a text line.
keyword	Name of a PRECMP directive. No blanks can occur between the master control character and the keyword; a comma or blank terminates the keyword. UPDATE keywords or their abbreviations are ignored.
p-list	Parameters for the PRECMP directive. A comma or blank(s) separates parameters in p-list. The parameters in p-list can be a FORTRAN expression. The parameters may include character strings enclosed by the apostrophe character as in ' ' or by the quote character as in " ". These are the only type of arguments that can contain embedded blanks. For directives *ROOTP, *ROOTN, and *ROOTD, the parameters may include numerical values enclosed by the left and right parenthesis characters as in ( ), the open and closed square brackets as in [ ], and the open and closed angle brackets as in < >. Parameters in p-list can include optional parameters which are specified in a sublist denoted by { }. Some directives have no parameters.
..	Continuation character for continuing input on the next line. However, any text after .. on the same line is considered to be a comment.
!	Optional end of statement character. Any text after ! on the same line is considered to be a comment.

PRECMP directives are defined for LCAP2 commands and parameters and are summarized in Sections 4.2.3 and 4.2.4, respectively. PRECMP directives are also defined for selecting the type of output produced by PRECMP. They are described in Section 4.2.5.

Descriptions of the LCAP2 commands are found in the Reference in Appendix A. The arguments of the parameter list, p-list, for LCAP2 parameters can be any of the following six types shown in Figure 4.2. (The first one is an argument which is not enclosed by special characters.)

Table 4.2: Types of P-list Arguments

	A number (REAL or INTEGER) or a FORTRAN expression
' '	Character string for entering FORTRAN character string
" "	Character string for writing out comments to the output file.
(,)	Pair of complex number in (real,imag) format for entering root information
[ ]	Pair of complex number in [zeta,omega] format for entering root information
< >	Real number in <tau> format for entering root information

The character string enclosed by the double quote "..." is converted to a FORTRAN print statement. This argument type is defined to aid the user in generating comments to the output file. It is an optional argument for all PRECMP directives but is omitted from the formal descriptor for each directive for the sake of brevity. This argument type is not positional and may appear more than once on a directive statement.

Examples of several types of p-list and p-list arguments are given in the next section.

4.2.2 PRECMP Examples

Examples are presented to describe several types of p-list and p-list arguments used by the PRECMP directives.

Example 4.1

Directive: *SPADD i j k

Sample Usage: `*SPADD 5 N1 3`

Code Generated: `CALL SPADD(5,N1,3)`

LCAP2 Function: $\mathbf{SPTF}_5 = \mathbf{SPTF}_{n1} + \mathbf{SPTF}_3$

Example 5.2

Directive: *SZXFM i j { sampt delay zoh }

note: { ... } designates optional parameter list

Sample Usage: `*SZXFM 4 6 .5`

Code Generated:

```
SAMPT=.5
CALL SZXFM(4,6)
```

note: since optional values are not specified for delay and zoh, previous or default values are assumed

LCAP2 Function: $\mathbf{ZPTF}_4$ = z transform of $\mathbf{SPTF}_6$

Example 4.3

Directive: *SZXFM i j { sampt delay zoh }

Sample Usage: `*SZXFM "Z-TRANS OF PLANT" 4 6 .5 .1*SAMPT`

Code Generated:

```
PRINT*,'Z-TRANS OF PLANT'
SAMPT=.5
DELAY=.1*SAMPT
CALL SZXFM(4,6)
```

Example 4.4

Directive: *SAVE file_name iprn

Sample Usage: `*SAVE 'DATA1' 1 "DATA FOR CONFIG 3X"`

Code Generated:

```
PRINT*,'DATA FOR CONFIG. 3X'
CALL SAVE('DATA1',1)
```

Example 4.5

Directive: *ROOTP {'HIGH'} gain r-list

r-list is a root list whose arguments can be any of the following types:

a — for real root in $(\frac{s}{-a} + 1)$ factored form. If a = 0, the factored form is (s).

(a,b) for complex root pair in $(\frac{s}{-a-jb}+1)(\frac{s}{-a+jb}+1)$ factored form

[zeta,omega] for complex root pair in $(\frac{s^2}{omega^2}+\frac{2\,zeta\,s}{omega}+1)$ factored form. If omega=0, the factored form is (s^2).

<tau> for real root in $(tau\ s+1)$ factored form. If tau=0, the root is not defined.

Sample Usage: *ROOTP 100. -1 (-2.,3.) [X,Y] <TAU> 0.

This directive will enter root data into the array ROOTP for the following polynomial

$$(100.)(s+1)(\frac{s}{2-j3}+1)(\frac{s}{2+j3}+1)(\frac{s^2}{Y^2}+\frac{2\,X\,s}{Y}+1)(TAUs+1)(s)$$

The actual code generated is a series of FORTRAN subroutine calls which (1) converts the various root formats of r-list into the appropriate one used by array POLYP, (2) counts the number of actual roots to be entered, and (3) convert, if necessary, a "high order gain" into a "low order gain" which is required for entering root data. The details on the code generated by the *ROOTP directive are given in Section 4.2.4.

Example 4.6

Directive: *ROOTP {'HIGH'} gain r-list

Sample Usage: *ROOTP 'HIGH' 200. -1 (-2.,3.) [X,Y] <TAU> 0.

This directive will enter root data into the array ROOTP for the following polynomial

$$(200.)(s+1)(s+2-j3)(s+2+j3)(s^2+2X*Ys+Y^2)(s+\frac{1}{TAU})(s)$$

See Section 4.2.4 for the details on how the high order coefficient, 200, is converted to the low order coefficient used to store root data in LCAP2.

4.2.3 Summary of LCAP2 PRECMP Command Directives

The following is an alphabetized list of the PRECMP command directives. Description of these directives is given in Appendix A.

```
*B1CEQ label indx isptf iyin nycin
*B1EIG i
*B1END
*B1FREQ
*B1INIT label oldnew ncblk
*B1LOAD file_name
*B1LOCI indx
*B1SAVE file_name
*B1TF i
*B1TIME
*B2CEQ label indx isptf delay iyin nycin nxsin
*B2DEQ label indx izptf dsampt delay iyin nycin nydin nxsin
*B2EIG i
*B2FREQ
*B2END
*B2INIT label oldnew ncblk ndblk nshblk
*B2LOAD file_name
*B2LOCI blktyp indx
*B2SAVE file_name
*B2SEQ label indx ssampt delay nydin nxsin
*B2TF i
*B2TIME
*CPYPS i j k
*CPYPW i j k
*CPYPZ i j k
*CPYSP i j k
*CPYWP i j k
*CPYZP i j k
*DETRM i { mxm mdeg }
*DTERM i j { mxm mdeg }
*LOAD file_name iprn
*PADD i j k
*PDEL i
*PEQU i j
*PLDC i
*PLDR i
*PMPY i j k
*POLY i
*PRTS i
*PSUB i j k
*SAVE file_name iprn
*SELCR i
*SFREQ i
```

```
*SLOCI i
*SNORM i { nrmhi knorm }
*SPADD i j k
*SPCLSLP i j k
*SPDEL i
*SPDIV i j k
*SPEQU i j
*SPLDC i
*SPLDR i
*SPMPY i j k
*SPRTS i
*SPSUB i j k
*SPTF i
*STIME i { ttype tmagn tzero tend tdelt }
*SWMRX i j { sampt delay zoh ntger }
*SWXFM i j { sampt delay zoh }
*SZMRX i j { sampt delay zoh ntger }
*SZXFM i j { sampt delay zoh }
*WELCR i
*WFREQ i { sampt }
*WLOCI i
*WMRFQ i { sampt ntger }
*WMRXFM i j { sampt ntger }
*WNORM i { nrmhi knorm }
*WPADD i j k
*WPCLSLP i j k
*WPDEL i
*WPDIV i j k
*WPEQU i j
*WPLDC i
*WPLDR i
*WPMPY i j k
*WPRTS i
*WPSUB i j k
*WPTF i
*WSXFM i j { sampt }
*WZXFM i j
*ZELCR i
*ZFREQ i { sampt }
*ZLOCI i
*ZMRFQ i { sampt ntger }
*ZMRXFM i j { sampt ntger }
*ZNORM i { nrmhi knorm }
*ZPADD i j k
*ZPCLSLP i j k
*ZPDEL i
*ZPDIV i j k
*ZPEQU i j
```

*ZPLDC i
*ZPLDR i
*ZPMPY i j k
*ZPRTS i
*ZPSUB i j
*ZPTF i
*ZSXFM i i { sampt }
*ZTIME i { sampt ttype tmagn tend }
*ZVCNG i j { ntger }
*ZWXFM i j

4.2.4 Summary of LCAP2 PRECMP Parameter Directives

PRECMP parameter directives are used to simplify the input of LCAP2 parameters which are arrays. Except for the directives, *ROOTP, *ROOTN, and *ROOTD, all PRECMP directives for LCAP2 parameter arrays will generate a sequence of FORTRAN statements to load the arguments of the directive into an array starting in element one. For example, the directive:

```
*POLYN 2 W1*W1 2.*Z1*W1 1
```

will generate the following FORTRAN code,

```
POLYN(1)=2
POLYN(2)=W1*W1
POLYN(3)=2.*Z1*W1
POLYN(4)=1
```

The following is a list of PRECMP directives for LCAP2 parameters which are arrays.

*FREQ1 fstart fend fdelta	"see command SFREQ in Appendix A" for
*FREQ2 fstart fend fdelta	definition of fstart, fend, and fdelta"
*FREQ3 fstart fend fdelta	
*FREQ4 fstart fend fdelta	
*FREQ5 fstart fend fdelta	
*KGAIN g_1 g_2 ... g_n	"where $n \leq 25$"
*IYCIN i_1 i_2 ... i_n	"where $n \leq$ MDIMYC or NDIMYC, see command B1CEQ or B2CEQ in Appendix A"
*IYDIN i_1 i_2 ... i_n	"where $n \leq$ NDIMYD, see command B2DEQ in Appendix A"
*IXSIN i_1 i_2 ... i_n	"where $n \leq$ NDIMXS, see command B2SEQ in Appendix A"
*POLYP ndeg a_0 a_1 ... a_{ndeg}	"where $ndeg \leq$ MXPDEG
*POLYD ndeg a_0 a_1 ... a_{ndeg}	"
*POLYN ndeg a_0 a_1 ... a_{ndeg}	"
*OMEGA ω_1 ω_2 ... ω_{nomega}	"where $nomega \leq 20$"
*ROOTP {'HIGH'} gain r-list	"see next paragraph"
*ROOTD {'HIGH'} gain r-list	"
*ROOTN {'HIGH'} gain r-list	"

In Section 6.1.2, the format used in LCAP2 for storing root data into a complex array is presented. For a polynomial with n roots, the roots are to be stored in elements 2 through n+1 of a complex array. The number of roots, n, is to be stored in the real part of element 1. The imaginary part of element 1 is used to store a "gain" which can be used to define a polynomial with the same set of roots. This "gain" is defined in LCAP2 as the low order nonzero coefficient of the polynomial.

In general, the root data which the analyst will be working with will not always be in this format. The PRECMP directives *ROOTP, *ROOTD, and *ROOTN will enable the user to enter root data with a more general format by converting the user data into the LCAP2 format described above. The PRECMP precompiler will do this conversion by generating a series of calls to subroutines RTREAL, RTCPLX, RTZOMG, RTTAU and RTGAIN. It also uses temporary variable IZZZZ to keep track of the number of roots that are to be entered. The code that is written by PRECMP to implement this conversion is described for the *POLYP directive below[1]:

1. After the *POLYP keyword has been detected, read the next argument after the keyword.
   - If it is 'HIGH', set an internal flag to indicate that the "gain" that will be entered later will be the "high order coefficient" instead of the "low order nonzero coefficient". Then go to Step 2.
   - If it is not 'HIGH', go to Step 3.
2. Read the next argument.
3. Write the following FORTRAN statements:

```
ZZGAIN = current argument
IZZZZ = 0                                   "initialize counter"
```

4. Read the next argument.
   - If there are no more arguments, go to Step 9.
   - If the argument is of the form, a , go to Step 5.
   - If the argument is of the form, (a,b) , go to Step 6.
   - If the argument is of the form, [zeta,omega] , go to Step 7.
   - If the argument is of the form, <tau> , go to Step 8.
5. Write the following FORTRAN statements and then go to Step 4.

```
ZZREAL = a                                  "where a is from the current argument"
CALL RTREAL(ROOTP,IZZZZ,ZZREAL)             "IZZZZ will be incremented by 1"
```

6. Write the following FORTRAN statements and then go to Step 4.

```
ZZREAL = a                                  "where a is from the current argument"
ZZIMAG = b                                  "where b is from the current argument"
CALL RTCPLX(ROOTP,IZZZZ,ZZREAL,ZZIMAG)      "IZZZZ will be incremented by 2"
```

[1] The code implemented for the *ROOTD and *ROOTN directives is similar to *ROOTP except that ROOTD or ROOTN is substituted for ROOTP.

7. Write the following FORTRAN statements and then go to Step 4.

```
ZZZETA = zeta                            "where zeta is from the current argument"
ZZOMEG = omega                           "where omega is from the current argument"
CALL RTZOMG(ROOTP,IZZZZ,ZZZETA,ZZOMEG)   "IZZZZ will be incremented by 2"
```

8. Write the following FORTRAN statements and then go to Step 4.

```
ZZTAU = tau                              "where tau is from the current argument"
CALL RTREAL(ROOTP,IZZZZ,ZZTAU)           "IZZZZ will be incremented by 1"
```

9. Write one of the following FORTRAN statement which completes the "cracking" of the *ROOTP directive.

```
CALL RTGAIN(ROOTP,IZZZZ,0,ZZGAIN)        "if 'HIGH' was not entered"
CALL RTGAIN(ROOTP,IZZZZ,1,ZZGAIN)        "if 'HIGH' was entered"
```

In Steps 5 through 8, the calls to subroutines RTREAL, RTCPLX, RTZOMG, or RTTAU will convert the current argument to the proper root format expected by LCAP2 and store it into ROOTP(IZZZZ+2). These subroutines will then increment IZZZZ either by 1 or 2. In Step 9 subroutine RTGAIN will set IZZZZ, the number of roots begin entered, into the real part of ROOTP(1). An internal flag will be checked to see if ZZGAIN is the low order nonzero coefficient or the high order coefficient of the polynomial. If the argument is zero, the imaginary part of ROOTP(1) will be set equal to ZZGAIN. If it is not, the high order coefficient will be converted to the low order nonzero coefficient and stored into the imaginary part of ROOTP(1). This conversion is computed as:

low order nonzero coefficient = ZZGAIN * product of (- nonzero roots)

4.2.5 PRECMP Directives for Selecting Type of Output

There are four PRECMP directives for selecting the type of output produced by PRECMP. The first two, *NLIST and *NOUTPUT, must be invoked at the beginning of the input file. The last two, *SUPRESS and *NSUPRESS, can be invoked anywhere in the input file.

Directive: *NLIST

By default, all the input file (card images) to PRECMP will be printed to the output file with a sequence number. However, if the *NLIST directive is used, the input file is not printed out. Unless the input is exceptionally long and the job is a production run, use of this directive is not recommended.

Directive: *NOUTPUT

PRECMP will create file PCMPILE which contains (1) all of the users' input data for program UPDATE and (2) FORTRAN statements generated from "cracking" the PRECMP directives. PCMPILE is the input file for the FORTRAN compiler. By default, the file PCMPILE will be printed to the output file. However, since most users will be compiling the FORTRAN code with the list option, there is no need to print out the file PCMPILE separately. Thus, the *NOUTPUT directive should normally be used.

Directive: *SUPRESS

When PRECMP "cracks" a PRECMP directive, it first creates FORTRAN comment statement(s) which contains a copy of the PRECMP directive so that the output of a FORTRAN compilation will show where the PRECMP directive was invoked. Then it will generate FORTRAN statements which will implement the PRECMP directive. In some cases the code generated may be quite long and of no interest to the typical user. The *SUPRESS directive will suppress the listing of these PRECMP-generated FORTRAN codes in the output of the FORTRAN compilation. It does this by using the following FORTRAN compiler directives that control the listing of its output:

On the CRAY the directive to stop the listing is: `CDIR$ NOLIST`

On the CDC the directive to stop the listing is: `C      LIST(S=0)`

Since only the PRECMP generated FORTRAN code is to be suppressed when the *SUPRESS directive is invoked, the FORTRAN compiler must be directed to resume listing after each set of PRECMP FORTRAN code is generated. The FORTRAN compiler directives to start the listing are:

On the CRAY the directive to start the listing is: `CDIR$ LIST`

On the CDC the directive to start the listing is: `C      LIST(S=1)`

Thus, when a PRECMP directive is being "cracked", PRECMP will check to see if *SUPRESS has been invoked. If it has, it will first write out a FORTRAN compiler directive to file PCMPILE to stop the listing. Then after the last PRECMP-generated FORTRAN code has been written for that particular directive, a FORTRAN compiler directive is written to file PCMPILE to restart the listing.

Directive: *NSUPRESS

The *NSUPRESS directive is used to negate the *SUPRESS directive so that all PRECMP-generated code will be listed by the FORTRAN compiler. Use of *SUPRESS and *NSUPRESS directives will enable the user to selectively suppress the listing of PRECMP generated output.

Chapter 5

Job Setup

A job setup consists of a set of control cards and an input file. Since the use of the preprocessor program PRECMP is optional[1] and batch LCAP2 can be run on either the CRAY, CDC MFB, or CDC MFX computers, there are six different typical setups that can be used. They are all available to the user as an indirect file on CDC INTERCOM by typing the following command:

```
IGET,file_name/PF=ZL2USER,ID=9487
```

where **file_name** and its description is given below in Table 5.1.

Table 5.1: Sample Deck Setups

Sample Deck Setups	
file_name	Description
CL2CC	For CRAY Without PRECMP
CL2CCPR	For CRAY With PRECMP
BL2CC	For CDC MFB Without PRECMP
BL2CCPR	For CDC MFB With PRECMP
XL2CC	For CDC MFX Without PRECMP
XL2CCPR	For CDC MFX With PRECMP

Each of these setups includes all the necessary control cards for generating plots. The control card for compilation includes the debug option. Not included are the control cards needed for attaching and saving files associated with the restart capability. That will be discussed in another part of the manual. Currently there are two versions of LCAP2 on the CRAY, a regular and a large model version. Sections 5.1 and 5.2 describe how they are selected.

[1] See Section 4.2

5.1 Setup for CRAY without PRECMP

The following is a typical deck setup CL2CC. Two different versions of the LCAP2 source and binary libraries are available. They are selected by the ID parameter on the ACCESS statements for files CLCAP2PL and CLCAP2LIB. In the control cards below, the regular version is selected by ID=MX49 (max. polynomial=49). For the large model version of LCAP2, use ID=MX100 (max. polynomial=100).

```
      ... (job card) i.e., JOB,JN=...,US=...,MFL=700000,T=30.
      ... (account card) ...
      ... (PRT card) ...
*
*FILE CL2CC, (IGET CL2CC/PF=ZL2USER,ID=9487)
*SAMPLE CONTROL CARD SETUP FOR CRAY WITHOUT PRECOMPILER
*
ACCESS,DN=$PL,PDN=CLCAP2PL,OWN=9487,ID=MX49.
UPDATE,CD,ED,ID,IN.
CFT,I=$CPL,ON=IX,DEBUG.
ASSIGN,DN=DAPOLY,U,A=FT84.
ASSIGN,DN=DASPTF,U,A=FT85.
ASSIGN,DN=DAWPTF,U,A=FT86.
ASSIGN,DN=DAZPTF,U,A=FT87.
ACCESS,DN=$OBL,PDN=CLCAP2LIB,OWN=9487,ID=MX49.
ACCESS,DN=$PLOT,PDN=ASCPLOTLIB,OWN=PUBLIC.
SEGLDR,GO,CMD='BIN=$BLD;LIB=$OBL,$PLOT;MAP=PART'.
HARDCPY,ST=IBMD8.
EXIT.
DUMPJOB.
DEBUG.
*EOR
*IDENT XXXXX
      PROGRAM LCAP2(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
*CALL COMLCAP2
************ INSERT ANY NON EXECUTABLE STATEMENTS HERE ****************
      CALL INITO
      CALL MINITO
C************** BEGIN USER CODE HERE FOR MAIN PROGRAM *****************
C         .
C     user code
C         .
C**************** END OF USER CODE FOR MAIN PROGRAM *******************
      CALL LEXIT
      END
```

5.2 Setup for CRAY with PRECMP

The following is a typical deck setup file CL2CCPR. Two different versions of PRECMP and the LCAP2 source and binary libraries are available. They are selected by the ID parameter on the ACCESS statements for files CLCAP2PRECMP, CLCAP2PL and CLCAP2LIB. In the control cards below, the regular version is selected by ID=MX49 (max. polynomial=49). For the large model version of LCAP2, use ID=MX100 (max. polynomial=100).

```
      ... (job card) i.e., JOB,JN=...,US=...,MFL=700000,T=30.
      ... (account card) ...
      ... (PRT card) ...
*
*FILE CL2CCPR, (IGET CL2CCPR/PF=ZL2USER,ID=9487)
*SAMPLE CONTROL CARD SETUP FOR CRAY WITH PRECOMPILER
*
ACCESS,DN=X,PDN=CLCAP2PRECMP,OWN=9487,ID=MX49.
X.
REWIND,DN=PCMPLIE.
IF(JO.EQ.'BAD'L)
    *---->FATAL ERROR IN PRECMP
    COPYF,I=PMCPILE,O=$OUTPUT,S=1.
    EXIT.
ENDIF.
ACCESS,DN=$PL,PDN=CLCAP2PL,OWN=9487,ID=MX49.
UPDATE,I=PCMPILE,CD,ED,ID,IN.
CFT,I=$CPL,ON=IX,DEBUG.
ASSIGN,DN=DAPOLY,U,A=FT84.
ASSIGN,DN=DASPTF,U,A=FT85.
ASSIGN,DN=DAWPTF,U,A=FT86.
ASSIGN,DN=DAZPTF,U,A=FT87.
ACCESS,DN=$OBL,PDN=CLCAP2LIB,OWN=9487,ID=MX49.
ACCESS,DN=$PLOT,PDN=ASCPLOTLIB,OWN=PUBLIC.
SEGLDR,GO,CMD='BIN=$BLD;LIB=$OBL,$PLOT;MAP=PART'.
HARDCPY,ST=IBMD8.
EXIT.
DUMPJOB.
DEBUG.
*EOR
*NOUTPUT
*IDENT XXXXX
*INSERT START.1
      PROGRAM LCAP2(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
*CALL COMLCAP2
************ INSERT ANY NON EXECUTABLE STATEMENTS HERE ****************
      CALL INITO
      CALL MINITO
C************** BEGIN USER CODE HERE FOR MAIN PROGRAM *****************
```

```
C           .
C      user code
C           .
C**************** END OF USER CODE FOR MAIN PROGRAM *******************
      CALL LEXIT
      END
```

5.3 Setup for CDC MFB without PRECMP

The following is a typical deck setup from file BL2CC.

```
      ... (job card) i.e., D305,STMFB,P3000,EC60,T=70.
      ... (account card) ...
      ... (XMIT card) ...
COMMENT.
COMMENT. FILE BL2CC, (IGET BL2CC/PF=ZL2USER,ID=9487)
COMMENT. SAMPLE CONTROL CARD SETUP FOR CDC MFB WITHOUT PRECOMPILER
COMMENT.
ATTACH(OLDPL,8BLCAP2PL,ID=9487)
UPDATE.
FTN5(I=COMPILE,LO,PL,DB=PMD)
RETURN(OLDPL)
ATTACH(LCAPLIB,8BLCAP2LIB,ID=9487)
ATTACH(PLOTLIB,FTN5PLOTLIB)
RFL,EC=60.
LDSET(LIB=LCAPLIB/PLOTLIB)
LDSET(PRESET=ZERO)
LGO(*OP=AF)
REWIND,ZZZZZMP.
COPYBF,ZZZZZMP,OUTPUT.
LIBRARY(PLOTLIB)
HARDCPY,ST=IMBD8.
EXIT.
REWIND,ZZZZZMP.
COPYBF,ZZZZZMP,OUTPUT.
*EOR
*IDENT XXXX
*INSERT START.1
      PROGRAM LCAP2(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
*CALL COMLCAP2
************ INSERT ANY NON EXECUTABLE STATEMENTS HERE ****************
      CALL INITO
      CALL MINITO
C************** BEGIN USER CODE HERE FOR MAIN PROGRAM *****************
C           .
C      user code
```

```
C          .
C**************** END OF USER CODE FOR MAIN PROGRAM *******************
      CALL LEXIT
      END
```

5.4 Setup for CDC MFB with PRECMP

The following is a typical deck setup BL2CCPR.

```
        ... (job card) i.e., D305,STMFB,P3000,EC60,T=70.
        ... (account card) ...
        ... (XMIT card) ...
COMMENT.
COMMENT. FILE BL2CCPR, (IGET BL2CCPR/PF=ZL2USER,ID=9487)
COMMENT. SAMPLE CONTROL CARD SETUP FOR CDC MFB WITH PRECOMPILER
COMMENT.
ATTACH(X,8BLCAP2PRECMP,ID=9487)
X.
RETURN,X.
REWIND,PCMPILE.
IFE,FILE(TAPE50,AS),NOFILE.
REMARK.PRECMP SUCCESSFULLY EXECUTED
ELSE,NOFILE.
REMARK.---->FATAL ERROR IN PRECMP
COPYSBF,PCMPILE,OUTPUT.
ABORT.
ENDIF,NOFILE.
ATTACH(OLDPL,8BLCAP2PL,ID=9487)
REQUEST(NEWPL,*PF)
UPDATE(I=PCMPILE)
FTN5(I=COMPILE,LO,PL,DB=PMD)
RETURN(OLDPL)
ATTACH(LCAPLIB,8BLCAP2LIB,ID=9487)
ATTACH(PLOTLIB,FTN5PLOTLIB)
RFL,EC=60.
LDSET(LIB=LCAPLIB/PLOTLIB)
LDSET(PRESET=ZERO)
LGO(*OP=AF)
REWIND,ZZZZZMP.
COPYBF,ZZZZZMP,OUTPUT.
LIBRARY(PLOTLIB)
HARDCPY,ST=IBMD8.
EXIT.
REWIND,ZZZZZMP.
COPYBF,ZZZZZMP,OUTPUT.
*EOR
```

```
*NOUTPUT
*IDENT XXXX
*INSERT START.1
      PROGRAM LCAP2(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
*CALL COMLCAP2
************ INSERT ANY NON EXECUTABLE STATEMENTS HERE ****************
      CALL INITO
      CALL MINITO
C************** BEGIN USER CODE HERE FOR MAIN PROGRAM *****************
C         .
C     user code
C         .
C**************** END OF USER CODE FOR MAIN PROGRAM *******************
      CALL LEXIT
      END
```

5.5 Setup for CDC MFX without PRECMP

The following is a typical deck setup from file XL2CC.

```
      ... (job card) i.e., D305,STMFX,P3000,MS300000,ML40,T=30.
      ... (account card) ...
      ... (XMIT card) ...
COMMENT.
COMMENT. FILE XL2CC, (IGET XL2CC/PF=ZL2USER,ID=9487)
COMMENT. SAMPLE CONTROL CARD SETUP FOR CDC MFX WITHOUT PRECOMPILER
COMMENT.
ATTACH(OLDPL,8XLCAP2PL,ID=9487)
UPDATE.
FTN5(I=COMPILE,LO,PL,DB=PMD)
RETURN(OLDPL)
ATTACH(PLOTLIB,FTN5PLOTLIB)
ATTACH(LCAPLIB,8XLCAP2LIB,ID=9487)
LIBRARY(LCAPLIB,PLOTLIB)
LDSET(PRESET=ZERO)
LGO(*OP=AF)
REWIND,ZZZZZMP.
COPYBF,ZZZZZMP,OUTPUT.
LIBRARY(PLOTLIB)
HARDCPY,ST=IBMD8.
EXIT.
REWIND,ZZZZZMP.
COPYBF,ZZZZZMP,OUTPUT.
*EOR
*IDENT XXXX
*INSERT START.1
```

```
      PROGRAM LCAP2(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
*CALL COMLCAP2
************ INSERT ANY NON EXECUTABLE STATEMENTS HERE ****************
      CALL INITO
      CALL MINITO
C************** BEGIN USER CODE HERE FOR MAIN PROGRAM *****************
C          .
C     user code
C          .
C**************** END OF USER CODE FOR MAIN PROGRAM *******************
      CALL LEXIT
      END
```

5.6 Setup for CDC MFX with PRECMP

The following is a typical deck setup from file XL2CCPR.

```
      ... (job card) i.e., D305,STMFX,P3000,MS300000,ML40,T=30.
      ... (account card) ...
      ... (XMIT card) ...
COMMENT.
COMMENT. FILE XL2CCPR, (IGET XL2CCPR/PF=ZL2USER,ID=9487)
COMMENT. SAMPLE CONTROL CARD SETUP FOR CDC MFX WITH PRECOMPILER
COMMENT.
ATTACH(X,8XLCAP2PRECMP,ID=9487)
X.
RETURN,X.
REWIND,PCMPILE.
IFE,FILE(TAPE50,AS),NOFILE.
REMARK.PRECMP SUCCESSFULLY EXECUTED
ELSE,NOFILE.
REMARK.---->ERROR IN PRECMP
COPYSBF,PCMPILE,OUTPUT.
ABORT.
ENDIF,NOFILE.
ATTACH(OLDPL,8XLCAP2PL,ID=9487)
REQUEST(NEWPL,*PF)
UPDATE(I=PCMPILE)
FTN5(I=COMPILE,LO,PL,DB=PMD)
RETURN(OLDPL)
ATTACH(PLOTLIB,FTN5PLOTLIB)
ATTACH(LCAPLIB,8XLCAP2LIB,ID=9487)
LIBRARY(LCAPLIB,PLOTLIB)
LDSET(PRESET=ZERO)
LGO(*OP=AF)
REWIND,ZZZZZMP.
```

```
COPYBF,ZZZZZMP,OUTPUT.
LIBRARY(PLOTLIB)
HARDCPY,ST=IBMD8.
EXIT.
REWIND,ZZZZZMP.
COPYBF,ZZZZZMP,OUTPUT.
*EOR
*NOUTPUT
*IDENT XXXX
*INSERT START.1
      PROGRAM LCAP2(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
*CALL COMLCAP2
************ INSERT ANY NON EXECUTABLE STATEMENTS HERE ****************
      CALL INITO
      CALL MINITO
C************** BEGIN USER CODE HERE FOR MAIN PROGRAM *****************
C         .
C     user code
C         .
C**************** END OF USER CODE FOR MAIN PROGRAM *******************
      CALL LEXIT
      END
```

Chapter 6

Data Structures

Four different data structures are defined for use in LCAP2. They are (1) LCAP2 parameters, (2) LCAP2 polynomials, (3) LCAP2 transfer functions, and (4) transfer function connection blocks. LCAP2 parameters are FORTRAN variables in common blocks DBASE, POLYCM, ROOTCM, and MATRIX1 used for entering data values used by LCAP2 commands. LCAP2 polynomials and transfer functions are data structures which the user can access or reference by indices in an LCAP2 command Transfer function connection blocks are data structures which are used to connect transfer function blocks for automated transfer function analysis.

6.1 LCAP2 Parameters

LCAP2 parameters are FORTRAN variables in common block DBASE used for entering data. A complete list of these parameters and their definition is given in Appendix B. Most of them are nondimensioned REAL variables which have preset or default values. There are also dimensioned REAL and COMPLEX arrays used for loading (1) coefficient and root data into polynomials and transfer functions and (2) polynomial elements into a matrix for determinant evaluation. Descriptions of these arrays are given below.

6.1.1 POLYP, POLYN, and POLYD

These are REAL arrays in common block POLYCM used for representing polynomials in coefficient form. Polynomial data is stored in these arrays as follows:

```
element (1)  =  degree of the polynomial              (.LE.MXPDEG[1])
        (2)  =  coefficient of the **0 term
        (3)  =  coefficient of the **1 term
         ⋮
      (n+2)  =  coefficient of the ** n-th term
```

[1] MXPDEG=49 for regular version of LCAP2, =100 for large model version on the CRAY

POLYP is used for loading coefficient data into $\mathrm{POLY_i}$ which is defined in Section 6.2. POLYN and POLYD are used for loading numerator and denominator coefficient data into $\mathrm{SPTF_i}$, $\mathrm{ZPTF_i}$, or $\mathrm{WPTF_i}$ which are defined in Section 6.3.

6.1.2 ROOTP, ROOTN, and ROOTD

These are COMPLEX arrays in common block ROOTCM used for representing polynomials in root or factored form. Root data is stored in these arrays as follows:

$$\begin{aligned} \text{element } (1) &= \text{CMPLX(deg,gain)} && \text{where deg=degree of the polynomial} \\ & && \text{and deg.LE.MXPDEG} \\ & && \text{gain=GAIN defined below} \\ (2) &= \text{root}_1 \\ (3) &= \text{root}_2 \\ &\vdots \\ (\text{n+1}) &= \text{root}_\text{n} \end{aligned}$$

The quantity GAIN is defined by the following representation of a polynomial:

$$GAIN \cdot s^{\mu} \sum_{i=1}^{n-\mu} \left(\frac{s}{-a_i} + 1\right)$$

where μ = number of roots at the origin.

The choice of the factor $(\frac{s}{-a_i} + 1)$ as opposed to $(\frac{s}{a_i} + 1)$ allows the user to enter roots into an array using their actual root location. If the $(\frac{s}{a_i} + 1)$ form were used instead, the sign of each root would have to be reversed before entering it into the array. The gain as defined above is associated with the lowest order nonzero coefficient of the polynomial. It is analogous to the Bode gain of an s plane transfer function. If the $(\frac{s}{-a_i} + 1)$ term were defined as $(s + a_i)$ instead, the definition of GAIN would be equal to the highest order coefficient of the polynomial. In this case, this gain would be analogous to the root locus gain of an s plane transfer function. Although the format used for entering root data into ROOTP, ROOTN, and ROOTD does not allow direct entry of a root locus gain, there are PRECMP directives, *ROOTP, *ROOTN, and *ROOTD, which have options for entering in a "root locus" gain instead of a "Bode" gain. When this option is selected, these directives will convert the "root locus gain" into an equivalent "Bode" gain used for representing the roots in the format defined above.

6.1.3 M0, M1, M2, M3, and M4

These are REAL arrays in common block MATRIX1 which are used for representing the elements of a matrix M(s) whose elements are polynomials. The M(s) matrix is used for transfer function evaluation via Cramer's method in which the determinant of a polynomial matrix is to be computed. Polynomial data in coefficient form is stored in these M0, M1, M2, M3, M4 matrices as follows:

$$\begin{aligned}
\mathrm{M0(i,j)} &= \text{coefficient of } s^0 \text{ term in element (i,j)} \\
\mathrm{M1(i,j)} &= \text{coefficient of } s^1 \text{ term in element (i,j)} \\
&\vdots \\
\mathrm{M4(i,j)} &= \text{coefficient of } s^4 \text{ term in element (i,j)}
\end{aligned}$$

6.1.4 B0, B1, B2, B3, and B4

These are REAL arrays in common block MATRIX1 which are used for representing the elements of a vector **B**(s) whose elements are polynomials. The **B**(s) vector is used by the command DTERM for transfer function evaluation via Cramer's method in which the determinant of a polynomial matrix is to be computed. Polynomial data in coefficient form is stored in these B0, B1, B2, B3, B4 vectors as follows:

$$\begin{aligned}
\mathrm{B0(i)} &= \text{coefficient of } s^0 \text{ term in element (i)} \\
\mathrm{B1(i)} &= \text{coefficient of } s^1 \text{ term in element (i)} \\
&\vdots \\
\mathrm{B4(i)} &= \text{coefficient of } s^4 \text{ term in element (i)}
\end{aligned}$$

6.2 LCAP2 Polynomials

The LCAP2 polynomial data structure is $\mathbf{POLY}_i$ which is defined to be referenced or accessed by the user with an index. This allows the user to perform functional operations on a $\mathbf{POLY}_i$ with simple FORTRAN CALL statements. For example, to add $\mathbf{POLY}_2$ to $\mathbf{POLY}_3$ and store the resultant polynomial into $\mathbf{POLY}_1$, the following FORTRAN statement can be used:

```
CALL PADD(1,2,3)
```

The index of $\mathbf{POLY}_i$ can be any nonnegative integer up to 999. By definition $\mathbf{POLY}_0$ is defined to be a zero degree polynomial equal to one.

The most common way for the user to input data into a $\mathbf{POLY}_i$ is to use either the PLDC (polynomial load coefficient form) or the PLDR (polynomial load root form) command. An example using the PLDC command was given already in Section 4.2 in which the coefficient data is first loaded into the FORTRAN array POLYP and then subroutine PLDC called. If the root or factored form of the polynomial is to be loaded in, root information is first loaded into the FORTRAN array ROOTP and then subroutine PLDR called.

Although the coefficient and the root or factored form (with the appropriate gain) are equivalent mathematically, each form of representation has its advantages. For addition or subtraction, the coefficient form is required. For multiplication, either form of representation can be used although the root form yields the more accurate results since the product can be obtained by collecting the roots of the multiplier and the multiplicand. An internal flag is defined for each $\mathbf{POLY}_i$ which keeps track of whether the root form representation is available. All $\mathbf{POLY}_i$'s will have a coefficient representation by definition. If roots of a $\mathbf{POLY}_i$ were loaded in or were computed by another

LCAP2 command, the root flag would be set so that subsequent operations using $\mathbf{POLY}_i$ can make use of the fact that its roots are available.

6.3 LCAP2 Transfer Functions

The LCAP2 transfer function data structures are SPTF_i, ZPTF_i, and WPTF_i which are defined to be referenced or accessed by the user with an index. This allows the user to perform functional operations on these transfer functions with simple FORTRAN CALL statements. For example, to multiply $\mathbf{SPTF}_4$ by $\mathbf{SPTF}_5$ and store the resultant product into $\mathbf{SPTF}_2$, the following FORTRAN statement can be used:

```
CALL SPMPY(2,4,5)
```

$\mathbf{SPTF}_i$, $\mathbf{ZPTF}_i$, and $\mathbf{WPTF}_i$ are, respectively, s, z, and w plane transfer functions. The index of these transfer functions can be any non negative integer up to 999. By definition $\mathbf{SPTF}_0$, $\mathbf{ZPTF}_0$, and $\mathbf{WPTF}_0$ are defined to be the unity transfer function.

The most common way for the user to input data into an $\mathbf{SPTF}_i$ is to use either the SPLDC (s plane load coefficient form) or the SPLDR (s plane load root form) command. For example, to load the following transfer function

$$\frac{s+A}{s^3+9s^2+26s+24}$$

into $\mathbf{SPTF}_3$, the following FORTRAN statements can be used:

```
POLYN(1)=1          "degree of numerator"
POLYN(2)=A
POLYN(3)=1
POLYD(1)=3          "degree of denominator"
POLYD(2)=24
POLYD(3)=26
POLYD(4)=9
POLYD(5)=1
CALL SPLDC(3)
```

If PRECMP directives were used instead, the above FORTRAN statements can be written by PRECMP if the user uses the following PRECMP directives:

```
*POLYN 1 A 1
*POLYD 3 24 26 9 1
*SPLDC 3
```

Another example is to load the following transfer function

$$\frac{20(s+2)}{(s+3)(s+4)(\frac{s^2}{10^2}+\frac{2(.5)s}{10}+1)}$$

into SPTF$_3$. First rewrite the transfer function in the form

$$\frac{40(\frac{s}{2}+1)}{12(\frac{s}{3}+1)(\frac{s}{4}+1)(\frac{s^2}{10^2}+\frac{2(.5)s}{10}+1)}$$

The following FORTRAN statements will load in data.

```
ROOTN(1) = (1.,40.)          "REAL(ROOTN(1))=number of numerator roots"
ROOTN(2) = (-2.,0.)          "IMAG(ROOTN(1))=numerator gain"
ROOTD(1) = (4.,12.)          "REAL(ROOTD(1))=number of denominator roots"
ROOTD(2) = (-3.,0.)          "IMAG(ROOTD(1))=denominator gain"
ROOTD(3) = (-4.,0.)
ROOTD(4) = (-5.,-8.6602)     "roots of quadratics must be computed by the user"
ROOTD(5) = (-5.,8.6602)
CALL SPLDR(3)
```

If PRECMP directives were used instead, the above FORTRAN statements can be written by PRECMP if the user uses the following PRECMP directives:

```
*ROOTN 40.  -2.
*ROOTD 12.  -3.  -4.  [.5,10.]
*SPLDR 3
```

Note that when the *ROOTN and *ROOTD PRECMP directives are used, the number of roots do not have to be entered. It will be computed by PRECMP. In the *ROOTD directive, use of the [zeta,omega] form for entering roots (see Section 4.2.1) will automatically convert zeta and omega to the root form required by the FORTRAN array ROOTD.

Like the **POLY**$_i$ data type, each **SPTF**$_i$, **ZPTF**$_i$, and **WPTF**$_i$ has an internal flag associated with it to keep track of whether the root form representation is available. When adding two transfer functions this flag will be checked to see if the roots are available for both operands. If available, LCAP2 will first factor out any common roots from the denominators before adding the two operands so that the order of the resultant transfer function will not become redundantly large.

6.4 Transfer Function Connection Blocks

For automated transfer function analysis in which a system is modeled as a connection of transfer function blocks, three different types of connection blocks are defined. They are the **C**$_i$, **D**$_i$, and **S**$_i$ blocks which represent, respectively, connection blocks for continuous, discrete, and sample-hold blocks. An overview on how these blocks are used was given in Section 2.7. In the next chapter a complete description on how they are used will be presented.

Chapter 7

Automated Analysis of Systems Connected by Transfer Function Blocks

The automated method for analysis of systems modeled as a connection of transfer functions was implemented to (1) simplify the effort required to set up an analysis, (2) provide a general method for handling continuous-discrete multirate systems, and (3) utilize state space methods for improved numerical accuracy. For continuous systems, this automated method provides a more efficient method of analysis for complex problems. For continuous-discrete multirate systems, this automated method provides more than an efficient method of analysis. It is a general analysis procedure which can model any continuous-discrete multirate system represented as a connection of transfer functions. With classical discrete transform methods, there are no general methods for performing an analysis based on block diagram reduction methods. Each system must be handled on a case-by-case basis. For multirate systems, there can be configurations which are not amenable to analysis by the classical block diagram reduction method. Thus, the automated analysis method described in this chapter represents not only a more efficient method for analyzing control systems, it provides a method for analyzing continuous-discrete multirate systems which cannot otherwise be done by using classical methods[1].

The following is a summary of the advantages of using the automated analysis method:

- Ease of setting up an analysis.
  - No transfer function algebra required.
  - No complex node equations (or signal flow graphs) are required to be written by the user.
  - No conversion of s plane transfer functions to Laplace transformed differential equations required.
  - User specifies only the inputs to each transfer function connection block.
  - Program will automatically (1) connect all the transfer function blocks, (2) remove any nondynamic states, and (3) generate a state space representation.

[1] At the present time the automated method does not handle time delays. The classical transform method, however, can handle time delays of continuous dynamics.

  - For SISO transfer function evaluation and frequency or time response calculation, the user only has to define the input and output blocks.
  - For root locus, the gain to be varied does not have to be a loop gain. The gain of any continuous or discrete connection block can be varied.

- General analysis method for multirate systems connected by transfer functions.
  - Analyzes configurations which are not possible with classical discrete transform methods

- Improved computational results.
  - Uses QR and QZ methods for computing eigenvalues and generalized eigenvalues, respectively.
  - Uses Kalman-Bertram method for analysis of continuous-discrete multirate systems.

Before describing the automated analysis method, the user code for specifying the connection of the transfer functions will be presented. Familiarity with the mechanics of setting up the connection of the transfer function blocks will provide a better understanding of the operations involved in computing the state space representation of a system.

7.1 Input Code Required for Connecting Transfer Function Blocks

The first part of an automated transfer function analysis is the input code required for specifying the connection of the transfer function blocks and the loading or computing of transfer function data. This section will discuss only the input code for specifying the connection of the transfer functions. This is a separate procedure from the loading or computing of transfer function data[1]. For continuous systems, $\mathbf{SPTF}_i$ blocks are to be connected. For continuous-discrete multirate systems, $\mathbf{SPTF}_i$, $\mathbf{ZPTF}_i$, and sample-hold blocks are to be connected. $\mathbf{WPTF}_i$ blocks are not used since discrete systems in the Kalman-Bertram method are represented as difference equations. For flexibility in defining and in modifying the connection of transfer function blocks, a set of continuous, discrete, and sample-hold connection blocks, $\mathbf{C}_i$, $\mathbf{D}_i$, and $\mathbf{S}_i$,[2] respectively, are defined. Connection of a system is to be described in terms of these $\mathbf{C}_i$, $\mathbf{D}_i$, and $\mathbf{S}_i$ blocks instead of the $\mathbf{SPTF}_i$ and $\mathbf{ZPTF}_i$ transfer functions which contains the actual data. To specify the connection of transfer functions of a system, the following is required of the user:

- Assign a transfer function connection block to each transfer function.
- Enter the number of connection blocks.
- Enter a descriptive label for each connection block.
- Enter an SPTF or ZPTF identifier number (a pointer) for each continuous or discrete connection block.
- Enter the number of inputs to each connection block.
- Enter the inputs connected to each connection block.

[1] This procedure can be done before or after the loading or computing of transfer function data

[2] Sample-hold connection blocks are zero-order hold blocks

- For each discrete or sample-hold connection block, enter the sampling period.

The procedure for specifying the connection of transfer functions will be explained in the following subsections by the use of examples. The same two examples from the overview in Chapter 2 for a continuous system and a continuous-discrete multirate system will be used. The commands associated with the automated transfer function analysis procedure will have the prefix B1 or B2. B1 will be used for continuous systems and B2 will be used for continuous-discrete systems.

7.1.1 Continuous System

The code to specify the connection of the following continuous system in Figure 7.1 will be presented.

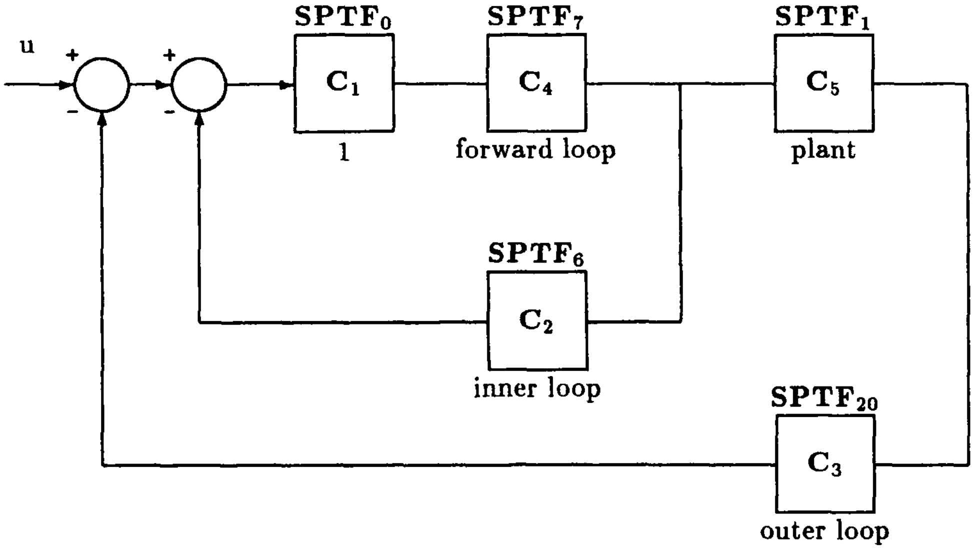


Figure 7.1: Continuous System Example

For batch LCAP2, three subroutines are used to specify the connection of a continuous system. They are (1) B1INIT for initialization, (2) B1CEQ for block equation definition, and (3) B1END for terminating the connection procedure. The procedure for specifying the connection always begins with a call to B1INIT and ends with a call to B1END. From the Reference in Appendix A for B1INIT, the call to B1INIT is described as:

FORTRAN CALL
```
CALL B1INIT( tlabel, oldnew, ncblk )
```

where, tlabel = label for block 1 connection (.LE.60 characters)
oldnew = 'OLD' for old data, (no initialization)
= 'NEW' for new data, (initializes all connections)
ncblk = number of s plane blocks

Each C_i block requires a separate call to B1CEQ. The call to B1CEQ is described as:

FORTRAN CALL CALL B1CEQ(label, indx, isptf, iyin, nycin, iycin)

where, label = label for block $\mathbf{C}_i$ (.LE.60 characters)
indx = block identifier for block $\mathbf{C}_i$ (1 - ncblk)
isptf = **SPTF** identifier for block $\mathbf{C}_i$ (0-999)
iyin = for future use. Enter a dummy value.
nycin = number of continuous blocks connected as inputs to block $\mathbf{C}_i$
iycin = array containing the identifiers of the continuous blocks connected to $\mathbf{C}_i$ (negative for sign change)

The FORTRAN code for this example can be written as:

```
      NCBLK=5
      CALL B1INIT('FIGURE 7.1 CONNECTION FILE','NEW',NCBLK)
      IYCIN(1)=-2
      IYCIN(2)=-3
      INDX=1
      ISPTF=0
      IYIN=0
      NYCIN=2
      CALL B1CEQ('SUMMING BLOCK USED TO DEFINE -(C2 + C3)'
     +,INDX,ISPTF,IYIN,NYCIN,IYCIN)
      IYCIN(1)=4
      CALL B1CEQ('INNER LOOP COMPENSATION',2,6,0,1,IYCIN)
      IYCIN(1)=5
      CALL B1CEQ('OUTER LOOP COMPENSATION',3,20,0,1,IYCIN)
      IYCIN(1)=1
      CALL B1CEQ('FORWARD LOOP COMPENSATION',2,7,0,1,IYCIN)
      IYCIN(1)=4
      CALL B1CEQ('PLANT',5,1,0,1,IYCIN)
      CALL B1END
```

Subroutine B1INIT will initialize the procedure for specifying the connection of the $\mathbf{C}_i$ blocks. In the first call to B1CEQ, use of assignment statements for arguments indx, isptf, and nycin were made for clarity. The actual inputs to $\mathbf{C}_1$ are specified by the values in IYCIN(1) and IYCIN(2) which, in this example, are both negative since there is negative feedback. Note that the input u is not included in the procedure for connecting the transfer functions. An input is only needed when a transfer function is evaluated. In the last four calls to B1CEQ, the arguments indx, isptf, and nycin were passed by value (instead of by name) to save a few lines of code. The B1END subroutine terminates the connection procedure. It will produce warning messages if incorrect data is entered or if a $\mathbf{C}_i$ block was not specified.

The above FORTRAN code can be equivalently written by the following PRECMP directives:

```
*B1INIT 'FIGURE 7.1 CONNECTION FILE' 'NEW' 5
*IYCIN -2 -3
*B1CEQ 'SUMMING BLOCK USED TO DEFINE -(C2 + C3)' 1 0 0 2
*IYCIN 4
```

```
*B1CEQ 'INNER LOOP COMPENSATION' 2 6 0 1
*IYCIN 5
*B1CEQ 'OUTER LOOP COMPENSATION' 3 20 0 1
*IYCIN 1
*B1CEQ 'FORWARD LOOP COMPENSATION' 2 7 0 1
*IYCIN 4
*B1CEQ 'PLANT' 5 1 0 1
*B1END
```

After the B1END subroutine is called, the following will be printed out to summarize the connection of the transfer functions.

```
NUMBER OF CONTINUOUS BLOCKS = 5
BLOCK     DESCRIPTION
C1        SUMMING BLOCK USED TO DEFINE -(C2 + C3)
C2        INNER LOOP COMPENSATOR
C3        OUTER LOOP COMPENSATOR
C4        FORWARD LOOP COMPENSATOR
C5        PLANT

BLOCK     TRANSFER FUNCTION           INPUTS FROM BLOCK
C1            SPTF0              -C2  ,-C3
C2            SPTF6               C4
C3            SPTF20              C5
C4            SPTF7               C1
C5            SPTF1               C4
```

The format of this printout enables the connections to be easily verified. A command B1SAVE is available which will save the connection data for a continuous system. For details and an example on its use, see the Reference in Appendix A and also Example 17 in Chapter 9. The transfer function connection data only describe the connection of the $\mathbf{C}_i$ blocks. They are totally independent of $\mathbf{SPTF}_i$ data except for the pointer or SPTF identifier number associated with each $\mathbf{C}_i$ block. This separation between connection of $\mathbf{C}_i$ data and $\mathbf{SPTF}_i$ data allows the user to model and analyze a system more efficiently. Revisions to an existing model can be made by substituting different SPTF identifiers for the appropriate $\mathbf{C}_i$ blocks.

In this example all the loops were closed. If an open forward loop transfer function were to be computed, the input to the $\mathbf{C}_2$ block would be set to zero and the transfer function between the input to $\mathbf{C}_2$ and the output of $\mathbf{C}_1$ would be computed. Without this automated transfer function connection capability, the effort to change a closed loop configuration to an open loop configuration can be significant if the system is complex.

The use of the unity[1] summing $\mathbf{C}_1$ block in this example was made so that an open loop transfer function can be computed. The command B1TF for transfer function evaluation requires that the desired output be an output of a $\mathbf{C}_i$ block. It cannot be a function of several $\mathbf{C}_i$ blocks. Thus, summing $\mathbf{C}_i$ blocks must be defined by the user for any variable of interest which is not an output of a $\mathbf{SPTF}_i$ block with dynamic states.

[1] By definition $\mathbf{SPTF}_0$ is equal to 1.

7.1.2 Continuous-Discrete Multirate Systems

The code to specify the connection of the following continous-discrete multirate system in Figure 7.2 will be presented.

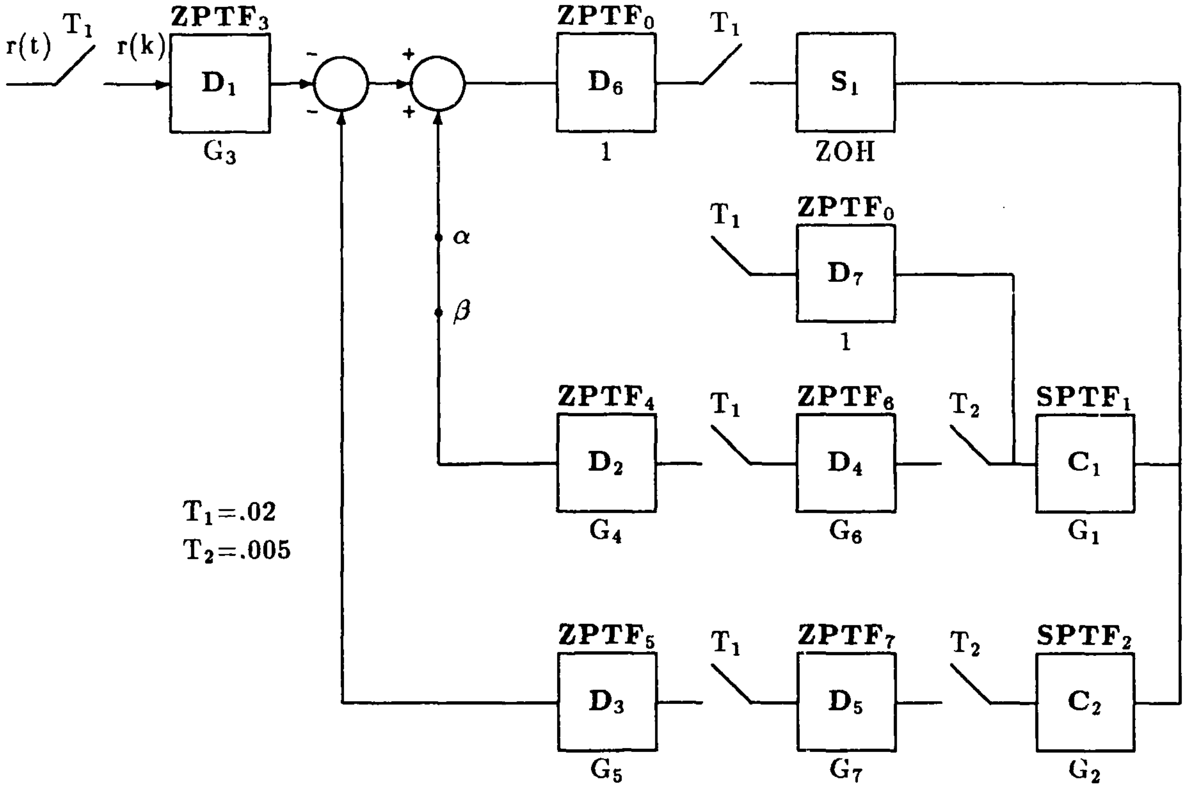


Figure 7.2: Continuous-Discrete Multirate System Example

For batch LCAP2, five subroutines are used to specify the connection of a continuous-discrete multirate system. They are (1) B2INIT for initialization, (2) B2CEQ for continuous block equation definition, (3) B2DEQ for discrete block equation definition, (4) B2SEQ for sample-hold equation definition, and (5) B2END for terminating the connection procedure. The procedure for specifying the connection always begins with a call to B2INIT and ends with a call to B2END. From the Reference in Appendix A for B2INIT, the call to B2INIT is described as:

FORTRAN CALL `CALL B2INIT( tlabel, oldnew, ncblk, ndblk, nshblk )`

where, tlabel = label for block 2 connection (.LE.60 characters)
oldnew = 'OLD' for old data, (no initialization)
= 'NEW' for new data, (initializes all connections)
ncblk = number of s plane blocks
ndblk = number of z plane blocks
nshblk = number of sample hold blocks

Each $\mathbf{C}_i$ block requires a separate call to B2CEQ. The call to B2CEQ is described as:

FORTRAN CALL

```
CALL B2CEQ( label, indx, isptf, delay, iyin, nycin, nxsin, iycin, ixsin )
```

where, label = label for block $\mathbf{C}_i$ (.LE.60 characters)
indx = block identifier for block $\mathbf{C}_i$ (1 - ncblk)
isptf = **SPTF** identifier for block $\mathbf{C}_i$
delay = delay of block $\mathbf{C}_i$, not implemented yet, but must enter a dummy value
iyin = for future use. Enter a dummy value.
nycin = number of continuous blocks connected as inputs to block $\mathbf{C}_i$
nxsin = number of sample-hold blocks connected as inputs to block $\mathbf{C}_i$
iycin = array containing the identifiers of the continuous blocks connected to $\mathbf{C}_i$. (negative for sign change)
ixsin = array containing the identifiers of the sample-hold blocks connected to $\mathbf{C}_i$. (negative for sign change)

Each $\mathbf{D}_i$ block requires a separate call to B2DEQ. The call to B2DEQ is described as:

FORTRAN CALL

```
  CALL B2DEQ( label, indx, izptf, dsampt, delay, iyin, nycin, nydin
+,nxsin, iycin, iydin, ixsin )
```

where, label = label for block $\mathbf{D}_i$ (.LE.60 characters)
indx = block identifier for block $\mathbf{D}_i$ (1 - ndblk)
izptf = **ZPTF** identifier for block $\mathbf{D}_i$
dsampt = sampling period of block $\mathbf{D}_i$
delay = delay of block $\mathbf{D}_i$, not implemented yet, but must enter a dummy value
iyin = for future use. Enter a dummy value.
nycin = number of continuous blocks connected as inputs to block $\mathbf{D}_i$
nydin = number of discrete blocks connected as inputs to block $\mathbf{D}_i$
nxsin = number of sample-hold blocks connected as inputs to block $\mathbf{D}_i$
iycin = array containing the identifiers of the continuous blocks connected to $\mathbf{D}_i$. (negative for sign change)
iydin = array containing the identifiers of the discrete blocks connected to $\mathbf{D}_i$. (negative for sign change)
ixsin = array containing the identifiers of the sample-hold blocks connected to $\mathbf{D}_i$. (negative for sign change)

Each $\mathbf{S}_i$ block requires a separate call to B2SEQ. The call to B2SEQ is described as:

FORTRAN CALL

```
CALL B2SEQ( label, indx, ssampt, delay, nydin, nxsin, iydin, ixsin )
```

where, label = label for block S_i (.LE.60 characters)
indx = block identifier for block S_i (1 - nshblk)
ssampt = sampling period of block S_i
delay = delay of block S_i, not implemented yet, but must enter a dummy value
nydin = number of discrete blocks connected as inputs to block S_i (See the Reference in Appendix A for constraint on sampling period of discrete block)
nxsin = number of sample-hold blocks connected as inputs to block S_i
iydin = array containing the identifiers of the discrete blocks connected to S_i. (negative for sign change)
ixsin = array containing the identifiers of the sample-hold blocks connected to S_i. (negative for sign change)

The FORTRAN code for the example in Figure 7.2 can be written as:

```
      NCBLK=2
      NDBLK=7
      NSHBLK=1
      CALL B2INIT('LABEL FOR EXAMPLE 2 CONNECTION FILE','NEW',NCBLK
     +,NDBLK,NSHBLK)
      IXSIN(1)=1
      INDX=1
      ISPTF=1
      DELAY=0.
      NYCIN=0
      NXSIN=1
      CALL B2CEQ('G1 BLOCK',INDX,ISPTF,DELAY,0,NYCIN,NXSIN,IYCIN,IXSIN)
      CALL B2CEQ('G2 BLOCK',2,2,0.,0,0,1,IYCIN,IXSIN)
      INDX=1
      IZPTF=3
      T1=.02
      DSAMPT=T1
      NYCIN=0
      NYDIN=0
      NXSIN=0
      CALL B2DEQ('G3 BLOCK',INDX,IZPTF,DSAMPT,DELAY,0,NYCIN,NYDIN,NXSIN
     +,IYCIN,IYDIN,IXSIN)
      IYDIN(1)=4
      CALL B2DEQ('G4 BLOCK',2,4,T1,DELAY,0,0,1,0,IYCIN,IYDIN,IXSIN)
      IYDIN(1)=5
      CALL B2DEQ('G5 BLOCK',3,5,T1,DELAY,0,0,1,0,IYCIN,IYDIN,IXSIN)
      T2=.005
      IYCIN(1)=1
      CALL B2DEQ('G6 BLOCK',4,6,T2,DELAY,0,1,0,0,IYCIN,IYDIN,IXSIN)
      IYCIN(1)=2
      CALL B2DEQ('G7 BLOCK',5,7,T2,DELAY,0,1,0,0,IYCIN,IYDIN,IXSIN)
      IYDIN(1)=-1
      IYDIN(2)=2
      IYDIN(3)=-3
      CALL B2DEQ('SUMMING BLOCK',6,0,T1,DELAY,0,0,3,0,IYCIN,IYDIN,IXSIN)
      IYCIN(1)=1
      CALL B2DEQ('SAMPLED OUTPUT OF G1',7,0,T1,DELAY,0,1,1,0,IYCIN,IYDIN,IXSIN)
      IYDIN(1)=6
      INDX=1
      SSAMPT=T1
      NYDIN=1
      NXSIN=0
      CALL B2SEQ('ZOH BLOCK',INDX,SSAMPT,DELAY,NYDIN,NXSIN,IYDIN,IXSIN)
      CALL B2END
```

Subroutine B2INIT will initialize the procedure for specifying the connection of the $\mathbf{C}_i$, $\mathbf{D}_i$, and $\mathbf{S}_i$ blocks. In the first calls to B2CEQ, B2DEQ, and B2SEQ, assignment statements for some of the arguments were used for clarity. In the remaining calls to these subroutines, most of the arguments were passed by value (instead of by name) to save a few lines of code. The B2END subroutine terminates the connection procedure. It will produce warning messages if incorrect data is entered or if a connection block was not specfied.

The above FORTRAN code can be equivalently written by the following PRECMP directives:

```
      NCBLK=2
      NDBLK=7
      NSHBLK=1
*B2INIT 'LABEL FOR EXAMPLE 2 CONNECTION FILE' 'NEW' NCBLK NDBLK NSHBLK
*IXSIN 1
      INDX=1
      ISPTF=1
      DELAY=0.
      NYCIN=0
      NXSIN=1
*B2CEQ 'G1 BLOCK' INDX ISPTF DELAY 0 NYCIN NXSIN
*B2CEQ 'G2 BLOCK' 2 2 DELAY 0 0 1
      INDX=1
      IZPTF=3
      T1=.02
      DSAMPT=T1
      NYCIN=0
      NYDIN=0
      NXSIN=0
*B2DEQ 'G3 BLOCK' INDX IZPTF DSAMPT DELAY 0 NYCIN NYDIN NXSIN
*IYDIN 4
*B2DEQ 'G4 BLOCK' 2 4 T1 DELAY 0 0 1 0
*IYDIN 5
*B2DEQ 'G5 BLOCK' 3 3 5 T1 DELAY 0 0 1 0
      T2=.005
*IYCIN 1
```

```
*B2DEQ 'G6 BLOCK' 4 6 T2 DELAY 0 1 0 0
*IYCIN 2
*B2DEQ 'G7 BLOCK' 5 7 T2 DELAY 0 1 0 0
*IYDIN -1 2 -3
*B2DEQ 'SUMMING BLOCK' 6 0 T1 DELAY 0 3 0 0
*IYCIN 1
*B2DEQ 'SAMPLED OUTPUT OF C1' 7 0 T1 DELAY 0 1 0 0
*IYDIN 6
      INDX=1
      SSAMPT=T1
      NYDIN=1
      NXSIN=0
*B2SEQ 'ZOH BLOCK' INDX SSAMPT DELAY NYDIN NXSIN
 CALL B2END
```

After the B2END subroutine is called, the following will be printed out to summarize the connection of the transfer functions.

```
NUMBER OF CONTINUOUS BLOCKS = 2
BLOCK     DESCRIPTION
C1        G1 BLOCK
C2        G2 BLOCK

BLOCK     TRANSFER FUNCTION          INPUTS FROM BLOCK
C1            SPTF1              S1
C2            SPTF2              S1
-----------------------------------
NUMBER OF DISCRETE BLOCKS = 7
BLOCK    DESCRIPTION
D1       G3 BLOCK
D2       G4 BLOCK
D3       G5 BLOCK
D4       G6 BLOCK
D5       G7 BLOCK
D6       SUMMING BLOCK
D7       SAMPLED OUTPUT OF C1

BLOCK    TRANSFER FUNCTION  SAMPLING PERIOD     INPUTS FROM BLOCKS
D1            ZPTF3          .20000E-01       HAS NO INPUT
D2            ZPTF4          .20000E-01       D4
D3            ZPTF5          .20000E-01       D5
D4            ZPTF6          .50000E-02       C1
D5            ZPTF7          .50000E-02       C2
D6            ZPTF0          .20000E-01      -D1  , D2  ,-D3
D7            ZPTF0          .20000E-01       C1
-----------------------------------
```

```
NUMBER OF SAMPLE-HOLD BLOCKS = 1
BLOCK    DESCRIPTION
S1       ZOH BLOCK

BLOCK    SAMPLING PERIOD            INPUTS FROM BLOCKS
S1          .20000E-01            D6
```

The format of this printout enables the connections to be easily verified. A command B2SAVE is available which will save the connection data for a continuous-discrete system. For details and an example on its use, see the Reference in Appendix A and also Example 18 in Chapter 9. The transfer function connection data only describe the connection of the $\mathbf{C}_i$, $\mathbf{D}_i$, and $\mathbf{S}_i$ blocks. They are totally independent of $\mathbf{SPTF}_i$ and $\mathbf{ZPTF}_i$ data except for the SPTF or ZPTF identifier number associated with each $\mathbf{C}_i$ or $\mathbf{D}_i$ block. This separation between connection of $\mathbf{C}_i$ and $\mathbf{D}_i$ data and $\mathbf{SPTF}_i$ and $\mathbf{ZPTF}_i$ data allows the user to model and analyze a system more efficiently. Revisions to an existing model can be made by substituting different SPTF or ZPTF identifiers for the appropriate $\mathbf{C}_i$ or $\mathbf{D}_i$ blocks.

In this example all the loops were closed. If an open loop transfer function at point P_1 in Figure 7.2 is to be determined, the connection from $\mathbf{D}_2$ to $\mathbf{D}_6$ must be broken and the transfer function between $\mathbf{D}_6$ and $\mathbf{D}_2$ calculated. Without this automated transfer function connection capability, the effort to change from a closed loop configuration to an open loop configuration would be significant.

The use of the unity[1] summing $\mathbf{D}_6$ block in this example was made so that an open loop transfer f nction can be computed. The command B2TF for transfer function evaluation requires that the desired output be an output of a $\mathbf{D}_i$ block. It cannot be a function of several $\mathbf{D}_i$ blocks. Thus, summing $\mathbf{D}_i$ blocks must be defined by the user for any variable of interest which is not an output of a $\mathbf{ZPTF}_i$ block with dynamic states.

The unity $\mathbf{D}_7$ block in this example was used so that the closed loop transfer function between R and the output of block "G_1" at the slower sampling rate can be computed with the command B2TF.

The complete statement and solution of this benchmark problem using the transfer function connection method is presented in Example 18 in Chapter 9.

[1] $\mathbf{ZPTF}_0$ is equal to the unity transfer function by definition

7.2 Derivation of State Space Description of Systems Connected by Transfer Function Blocks

The state space description of systems represented by a connection of transfer functions is solved for by using data from (1) the transfer function connection data specified by the procedure described in Section 7.1 and (2) **SPTF**$_i$ and **ZPTF**$_i$ data. To compute the state space representation of a system, the transfer function associated with each connection block is first transformed into a subsystem in state space form. Then using the connection data, the subsystems are combined to form a single state space representation. The derivation to be presented is for a single input since only SISO transfer functions are computed by LCAP2. Continuous and continuous-discrete multirate systems are presented separately.

7.2.1 Continuous Systems

Each transfer function associated with a $\mathbf{C}_i$ connection block is converted to the state space form

$$\begin{aligned} \dot{\mathbf{x}}_{ci} &= \acute{\mathbf{A}}_{ci}\mathbf{x}_{ci} + \acute{\mathbf{B}}_{ci}\acute{u}_c \\ y_{ci} &= \acute{\mathbf{C}}_{ci}\mathbf{x}_{ci} + \acute{\mathbf{D}}_{ci}\acute{u}_c \end{aligned} \tag{7.1}$$

The number of states in the vector $\mathbf{x}_{ci}$ is determined by the order of the s plane transfer function associated[1] with block $\mathbf{C}_i$. The output of block $\mathbf{C}_i$ is the scalar y_{ci}. The input $\acute{u}_c$ represents all the inputs into block $\mathbf{C}_i$. These inputs can only be outputs from other $\mathbf{C}_i$ blocks[2] and the single input u to the system. Note that when the transfer function associated with block $\mathbf{C}_i$ is a scalar, the matrices $\acute{\mathbf{A}}_{ci}$, $\acute{\mathbf{B}}_{ci}$, and $\acute{\mathbf{C}}_{ci}$ will be zero.

The state space representation for all the $\mathbf{C}_i$ blocks can be combined and expressed as

$$\begin{aligned} \dot{\mathbf{x}}_c &= \acute{\mathbf{A}}_c\mathbf{x}_c + \acute{\mathbf{B}}_c u + \mathbf{E}_c\mathbf{y}_c \\ \mathbf{y}_c &= \acute{\mathbf{C}}_c\mathbf{x}_c + \acute{\mathbf{D}}_c u + \mathbf{F}_c\mathbf{y}_c \end{aligned} \tag{7.2}$$

where

$$\begin{aligned} ncblk &= \text{number of } \mathbf{C}_i \text{ blocks} \\ mc &\leq ncblk \quad (< ncblk \text{ if there is a } \mathbf{C}_i \text{ block with no dynamics}) \\ \mathbf{x}_c &= \begin{bmatrix} \mathbf{x}_{c1} \\ \mathbf{x}_{c2} \\ \vdots \\ \mathbf{x}_{c\,mc} \end{bmatrix} \\ \mathbf{y}_c &= \begin{bmatrix} y_{c1} \\ y_{c2} \\ \vdots \\ y_{c\,ncblk} \end{bmatrix} \\ \acute{\mathbf{A}}_c &= \text{blockdiag}\left[\acute{\mathbf{A}}_{c1}, \acute{\mathbf{A}}_{c2}, \ldots, \acute{\mathbf{A}}_{c\,mc}\right] \end{aligned}$$

[1] Determined by the block identifier for $\mathbf{C}_i$.

[2] They are specified by the array IYCIN described in Section 7.1

$$
\begin{aligned}
\acute{\mathbf{B}}_c &= \begin{bmatrix} \acute{\mathbf{B}}_{c1} \\ \acute{\mathbf{B}}_{c2} \\ \vdots \\ \acute{\mathbf{B}}_{c\,mc} \end{bmatrix} \\
\acute{\mathbf{C}}_c &= \text{blockdiag}\left[\acute{\mathbf{C}}_{c1}, \acute{\mathbf{C}}_{c2}, \ldots, \acute{\mathbf{C}}_{c\,ncblk}\right] \\
\acute{\mathbf{D}}_c &= \begin{bmatrix} \acute{\mathbf{D}}_{c1} \\ \acute{\mathbf{D}}_{c2} \\ \vdots \\ \acute{\mathbf{D}}_{c\,ncblk} \end{bmatrix}
\end{aligned}
$$

$$
\mathbf{E}_c = \begin{bmatrix}
\acute{\mathbf{B}}_{c1} e_c(1,1) & \acute{\mathbf{B}}_{c1} e_c(1,2) & \cdots & \acute{\mathbf{B}}_{c1} e_c(1, ncblk) \\
\acute{\mathbf{B}}_{c2} e_c(2,1) & \acute{\mathbf{B}}_{c2} e_c(2,2) & & \\
\vdots & \vdots & \ddots & \\
\acute{\mathbf{B}}_{c\,mc} e_c(mc, 1) & & & \acute{\mathbf{B}}_{c\,mc} e_c(mc, ncblk)
\end{bmatrix}
$$

where $e_c(i,j)$ is equal to 1 if block $\mathbf{C}_\mathrm{j}$ is an input to block $\mathbf{C}_\mathrm{i}$; otherwise it is equal to zero.

$$
\mathbf{F}_c = \begin{bmatrix}
\acute{\mathbf{D}}_{c1} f_c(1,1) & \acute{\mathbf{D}}_{c1} f_c(1,2) & \cdots & \acute{\mathbf{D}}_{c1} f_c(1, ncblk) \\
\acute{\mathbf{D}}_{c2} f_c(2,1) & \acute{\mathbf{D}}_{c2} f_c(2,2) & & \\
\vdots & \vdots & \ddots & \\
\acute{\mathbf{D}}_{c\,ncblk} f_c(ncblk, 1) & & & \acute{\mathbf{D}}_{c\,ncblk} f_c(ncblk, ncblk)
\end{bmatrix}
$$

where $f_c(i,j)$ is equal to 1 if block $\mathbf{C}_\mathrm{j}$ is an input to block $\mathbf{C}_\mathrm{i}$; otherwise it is equal to zero.

Solving for the vector $\mathbf{y}_c$, the system can be written as

$$
\begin{aligned}
\dot{\mathbf{x}}_c &= \mathbf{A}_c \mathbf{x}_c + \mathbf{B}_c u \\
\mathbf{y}_c &= \mathbf{C}_c \mathbf{x}_c + \mathbf{D}_c u
\end{aligned}
\qquad (7.3)
$$

where

$$
\begin{aligned}
\mathbf{C}_c &= [\mathbf{I} - \mathbf{F}_c]^{-1}\, \acute{\mathbf{C}}_c \\
\mathbf{D}_c &= [\mathbf{I} - \mathbf{F}_c]^{-1}\, \acute{\mathbf{D}}_c \\
\mathbf{A}_c &= \acute{\mathbf{A}}_c + \mathbf{E}_c \mathbf{C}_c \\
\mathbf{B}_c &= \acute{\mathbf{B}}_c + \mathbf{E}_c \mathbf{D}_c
\end{aligned}
$$

Five LCAP2 commands are available for analysis based upon this state space representation. They are:

- B1EIG - Eigenvalues of the system matrix
- B1LOCI - Loci of the eigenvalues as a function of gain applied to any $\mathbf{C}_\mathrm{i}$ block
- B1TF - SISO transfer function evaluation in rational form between any two $\mathbf{C}_\mathrm{i}$ blocks

- B1FREQ - SISO frequency response between any two $\mathbf{C}_i$ blocks (without solving for the poles and zeros of the transfer function)

- B1TIME - SISO time response (by state space method) between any two $\mathbf{C}_i$ blocks

Although a frequency response can be computed by first evaluating a transfer function in rational form using the command B1TF and then using the frequency response command SFREQ, the command B1FREQ is implemented as an alternate method for computing the frequency response for the following reasons:

- If the poles and zeros cannot be computed accurately by command B1TF, any subsequent frequency response calculations using the command SFREQ on a transfer function evaluated with command B1TF will be limited to the accuracy of the poles and zeros. Although the limited experience in using the B1TF command on test and benchmark cases so far has not given any indications that the accuracy of the poles and zeros are in question, this concern must be addressed since this is a difficult numerical problem to solve in a reliable and accurate manner. This is particularly true for the zeros which are solved for by the QZ method.

- The frequency responses computed by command B1FREQ uses Laub's state space method [8] which does not involve the computation of the zeros and poles of the transfer function. This method should be able to compute the frequency response of systems represented in state space form which cannot be reliably and accurately computed by the combination of commands B1TF and SFREQ.

The command B1FREQ was implemented primarily as a means to enable the user to (1) check the accuracy of the rational transfer function computed by command B1TF[1] and (2) compute a frequency response should the B1TF command yield inaccurate results.

The two methods used by commands B1TF and B1FREQ for computing a transfer function are to be presented next. For command B1TF, the transfer function from block $\mathbf{C}_i$ to block $\mathbf{C}_j$ for an input u into block $\mathbf{C}_i$ is given by

$$\frac{y_j(s)}{u(s)} = \frac{\det \begin{bmatrix} s\mathbf{I} - \mathbf{A} & \mathbf{B} \\ \text{row j of } -\mathbf{C} & \text{row j of } \mathbf{D} \end{bmatrix}}{\det \begin{bmatrix} s\mathbf{I} - \mathbf{A} \end{bmatrix}} \tag{7.4}$$

where u is applied to block $\mathbf{C}_i$. Equation (7.4) is used for implementing the command B1TF since it can be used to compute the roots of the transfer function directly[2]. At the present time the QZ and QR methods are used to solve for the poles and zeros, respectively. The QZ method, though, is not always reliable and is sensitive to both (1) the order of the $\mathbf{SPTF}_i$ blocks associated with the $\mathbf{C}_i$ connection block and (2) the method used to convert from transfer function to state space representation. At the present time the observer canonical form is used to compute the state space representation. Future efforts to improve the reliability and accuracy of the B1TF command will include:

[1] By comparing frequency responses computed by the command B1FREQ and by the combination of commands B1TF and SFREQ.

[2] An alternate method is to first compute the coefficients of the determinants and then find the roots of the resultant polynomials. This is an inherently less accurate method for computing the roots.

- Evaluate both the resultant numerator and denominator polynomials at two different frequencies and compare the results with the evaluation of the two determinants in (7.4).
- Allow the user to select a root finding method[1] for solving the numerator and denominator roots instead of using the QR and QZ methods even though this method, in general, is less accurate.
- Use a partial fraction method for converting from transfer function to state space representation.

For command B1FREQ, the transfer function from block $\mathbf{C}_i$ to block $\mathbf{C}_j$ is given by j-th row of

$$\frac{\mathbf{Y}(s)}{\mathrm{u}(s)} = \mathbf{C}[\, s\mathbf{I} - \mathbf{A}\,]^{-1}\mathbf{B} + \mathbf{D} \tag{7.5}$$

Although a time response can be computed by first evaluating a transfer function in rational form using the command B1TF and then using the time response command STIME (which is based on an inverse Laplace transform method), the command B1TIME is implemented as an alternate method for computing the time response for the following reasons:

- The accuracy of the computed time response by command STIME depends on the accurate knowledge of the poles and zeros of a transfer function. As discussed previously, the poles and zeros computed by command B1TF may not always be solved in a reliable and accurate manner.
- The time response implemented by command B1TIME does not use the inverse Laplace transform method. It uses the solution of (7.3) evaluated at every T seconds given by

$$\begin{aligned} \mathbf{x}(kT+T) &= e^{\mathbf{A}T}\mathbf{x}(kT) + \int_0^T e^{\mathbf{A}\tau}d\tau\mathbf{B}u(kT) \\ \mathbf{y}(kT) &= \mathbf{C}\mathbf{x}(kT) + \mathbf{D}u(kT) \end{aligned} \tag{7.6}$$

The command B1TIME computes the time response from block $\mathbf{C}_i$ to block $\mathbf{C}_j$ where j is the j-th row of the output vector $\mathbf{y}$(kT) in (7.6).

The use of nondynamic $\mathbf{C}_i$ blocks in specifying the connection of transfer function blocks enables the user to model any variable of interest. Consideration, though, must be made on how these nondynamic or algebraic states will affect the subsequent computation of eigenvalues and transfer functions. In some control analysis programs[2] which have a similar block diagram connection capability to LCAP2, use of nondynamic gain blocks will, in general, increase the dimension of the system matrix. This of course will result in a nonminimal state space realization. For transfer function evaluation, this means that the added dimension of the system matrix will yield some redundant common poles and zeros. This higher dimensionality of the system matrix will require

[1] The same one used by commands DTERM and DETRM.

[2] PC-MATLAB and CTRL-C

more execution time for computing eigenvalues and transfer functions as well as the possibility of less accurate results.

In the formulation of the state space representation used by LCAP2[1], nondynamic $\mathbf{C}_i$ blocks will not affect the dimension of the system matrix. They only add to the the row dimension of matrices **C** and **D**. Thus, in the evaluation of a SISO transfer function using (7.4), nondynamic $\mathbf{C}_i$ blocks will not introduce unnecessary and redundant common roots in the numerator and denominator[2]. In practice an analyst will naturally want to model a system in terms of physical variables. This typically will require the use of many nondynamic $\mathbf{C}_i$ blocks. For example, in the IEEE Simple Continuous Model benchmark in Example 17 in Chapter 9, 12 out of the 21 $\mathbf{C}_i$ blocks are nondynamic blocks. If a state space formulation in which the dimension of the system matrix was increased by one for each nondynamic $\mathbf{C}_i$ block were used instead of (7.2), the dimension of the system matrix would have been increased from 14 (the actual number of dynamic states) to 26.

7.2.2 Continuous-Discrete Multirate Systems

There are two parts to computing the state space representation of a continuous-discrete system modeled as a connection of transfer functions. The first is computing the state space representations for (1) the continuous and discrete dynamics using the connection data and the $\mathbf{SPTF}_i$ and $\mathbf{ZPTF}_i$ data and (2) the sample-hold states. The formulation will yield representations in which the elements of the output vectors $\mathbf{y}_c$ and $\mathbf{y}_d$ correspond to the outputs of the $\mathbf{C}_i$ and $\mathbf{D}_i$ blocks, respectively. The sample-hold states are functions of the vector $\mathbf{y}_d$.

For the continuous part of the system, each transfer function associated with a $\mathbf{C}_i$ connection block is converted to the state space form

$$\begin{aligned} \dot{\mathbf{x}}_{ci} &= \acute{\mathbf{A}}_{ci}\mathbf{x}_{ci} + \acute{\mathbf{B}}_{ci}\acute{u}_c \\ y_{ci} &= \acute{\mathbf{C}}_{ci}\mathbf{x}_{ci} + \acute{\mathbf{D}}_{ci}\acute{u}_c \end{aligned} \qquad (7.7)$$

The number of states in the vector $\mathbf{x}_{ci}$ is determined by the order of the s plane transfer function associated[3] with block $\mathbf{C}_i$. The output of block $\mathbf{C}_i$ is the scalar y_{ci}. The input $\acute{u}_c$ represents all the inputs into block $\mathbf{C}_i$. These inputs can only be outputs from other $\mathbf{C}_i$ blocks[4] and $\mathbf{S}_i$ blocks[5]. No inputs from a $\mathbf{D}_i$ block are allowed. Also no input u_c is allowed since transfer functions will only be defined between $\mathbf{D}_i$ blocks. Note that when the transfer function associated with block $\mathbf{C}_i$ is a scalar, the matrices $\acute{\mathbf{A}}_{ci}$, $\acute{\mathbf{B}}_{ci}$, and $\acute{\mathbf{C}}_{ci}$ will be zero.

The state space representation for all the $\mathbf{C}_i$ blocks can be combined and expressed as

$$\begin{aligned} \dot{\mathbf{x}}_c &= \acute{\mathbf{A}}_c\mathbf{x}_c + \acute{\mathbf{B}}_c\mathbf{x}_s + \mathbf{E}_c\mathbf{y}_c \\ \mathbf{y}_c &= \acute{\mathbf{C}}_c\mathbf{x}_c + \acute{\mathbf{D}}_c\mathbf{x}_s + \mathbf{F}_c\mathbf{y}_c \end{aligned} \qquad (7.8)$$

where

$$nshblk = \text{number of } \mathbf{S}_i \text{ blocks}$$

[1] Eq. (7.2).

[2] This does not imply that the transfer function is a minimal realization. There may be common poles and zeros that are a function only of the dynamics blocks that are connected.

[3] Determined by the block identifier for $\mathbf{C}_i$.

[4] They are specified by the array IYCIN described in Section 7.1.

[5] They are specified by the array IXSIN described in Section 7.1.

$$\mathbf{x}_s = \begin{bmatrix} \mathbf{x}_{s1} \\ \mathbf{x}_{s2} \\ \vdots \\ \mathbf{x}_{s\,nshblk} \end{bmatrix}$$

and $ncblk$, mc, $\mathbf{x}_c$, $\mathbf{y}_c$, $\acute{\mathbf{A}}_c$, $\acute{\mathbf{B}}_c$, $\acute{\mathbf{C}}_c$, $\acute{\mathbf{D}}_c$, $\mathbf{E}_c$, and $\mathbf{F}_c$ are defined in (7.2).

Solving for the vector $\mathbf{y}_c$, the system can be written as

$$\begin{aligned} \dot{\mathbf{x}}_c &= \mathbf{A}_c\mathbf{x}_c + \mathbf{B}_c\mathbf{x}_s \\ \mathbf{y}_c &= \mathbf{C}_c\mathbf{x}_c + \mathbf{D}_c\mathbf{x}_s \end{aligned} \tag{7.9}$$

where

$$\begin{aligned} \mathbf{C}_c &= [\mathbf{I} - \mathbf{F}_c]^{-1}\,\acute{\mathbf{C}}_c \\ \mathbf{D}_c &= [\mathbf{I} - \mathbf{F}_c]^{-1}\,\acute{\mathbf{D}}_c \\ \mathbf{A}_c &= \acute{\mathbf{A}}_c + \mathbf{E}_c\mathbf{C}_c \\ \mathbf{B}_c &= \acute{\mathbf{B}}_c + \mathbf{E}_c\mathbf{D}_c \end{aligned}$$

The discussion in Section 7.2.1 on how nondynamic $\mathbf{C}_\mathrm{i}$ blocks will nct affect the dimension of the system matrix applies to (7.9) as well.

For the discrete part of the system, each transfer function associated with a $\mathbf{D}_\mathrm{i}$ connection block is transformed to the state space form

$$\begin{aligned} \mathbf{x}_{di}(t_{k+i}) &= \acute{\mathbf{A}}_{di}\mathbf{x}_{di}(t_k) + \acute{\mathbf{B}}_{di}\acute{u}_d(t_k) \\ y_{di}(t_k) &= \acute{\mathbf{C}}_{di}\mathbf{x}_{di}(t_k) + \acute{\mathbf{D}}_{di}\acute{u}_d(t_k) \end{aligned} \tag{7.10}$$

The number of states in the vector $\mathbf{x}_{di}$ is determined by the order of the z plane transfer function associated[1] with block $\mathbf{D}_\mathrm{i}$. The output of block $\mathbf{D}_\mathrm{i}$ is the scalar y_{di}. The input $\acute{u}_d$ represents all the inputs into block $\mathbf{D}_\mathrm{i}$. These inputs can only be outputs from (1) $\mathbf{C}_\mathrm{i}$ blocks[2], (2) $\mathbf{D}_\mathrm{i}$ blocks[3], (3) $\mathbf{S}_\mathrm{i}$ blocks[4] and (4) the single input u_d to the system. Note that when the transfer function associated with block $\mathbf{D}_\mathrm{i}$ is a scalar, the matrices $\acute{\mathbf{A}}_{di}$, $\acute{\mathbf{B}}_{di}$, and $\acute{\mathbf{C}}_{di}$ will be zero.

The state space representation for all the $\mathbf{D}_\mathrm{i}$ blocks can be combined and expressed as

$$\begin{aligned} \mathbf{x}_d(t_{k+1}) &= \acute{\mathbf{A}}_d\mathbf{x}_d(t_k) + \acute{\mathbf{B}}_d u_d(t_k) + \acute{\mathbf{E}}_d\mathbf{y}_c(t_k) + \acute{\mathbf{G}}_d\mathbf{x}_s(t_k) + \mathbf{P}_d\mathbf{y}_d(t_k) \\ \mathbf{y}_d(t_k) &= \acute{\mathbf{C}}_d\mathbf{x}_d(t_k) + \acute{\mathbf{D}}_d u_d(t_k) + \acute{\mathbf{F}}_d\mathbf{y}_c(t_k) + \acute{\mathbf{H}}_d\mathbf{x}_s(t_k) + \mathbf{Q}_d\mathbf{y}_d(t_k) \end{aligned} \tag{7.11}$$

where

$$ndblk = \text{number of } \mathbf{D}_\mathrm{i} \text{ blocks}$$

[1] Determined by the block identifier for $\mathbf{D}_\mathrm{i}$.

[2] They are specified by the array IYCIN described in Section 7.1.

[3] They are specified by the array IYDIN described in Section 7.1.

[4] They are specified by the array IXSIN described in Section 7.1. The output of each sample-hold block is $\mathbf{x}_{si}$.

$$
\begin{aligned}
nshblk &= \text{number of } \mathbf{S}_\mathrm{i} \text{ blocks} \\
md &\le ndblk \;\; (< ndblk \text{ if there is a } \mathbf{D}_\mathrm{i} \text{ block with no dynamics}) \\
\mathbf{x}_d &= \begin{bmatrix} \mathbf{x}_{d1} \\ \mathbf{x}_{d2} \\ \vdots \\ \mathbf{x}_{d\,md} \end{bmatrix} \\
\mathbf{y}_d &= \begin{bmatrix} \mathbf{y}_{d1} \\ \mathbf{y}_{d2} \\ \vdots \\ \mathbf{y}_{d\,ndblk} \end{bmatrix} \\
\mathbf{x}_s &= \begin{bmatrix} \mathbf{x}_{s1} \\ \mathbf{x}_{s2} \\ \vdots \\ \mathbf{x}_{s\,nshblk} \end{bmatrix} \\
\acute{\mathbf{A}}_d &= \text{blockdiag}\left[\acute{\mathbf{A}}_{d1}, \acute{\mathbf{A}}_{d2}, \ldots, \acute{\mathbf{A}}_{d\,md}\right] \\
\acute{\mathbf{B}}_d &= \begin{bmatrix} \acute{\mathbf{B}}_{d1} \\ \acute{\mathbf{B}}_{d2} \\ \vdots \\ \acute{\mathbf{B}}_{d\,md} \end{bmatrix} \\
\acute{\mathbf{C}}_d &= \text{blockdiag}\left[\acute{\mathbf{C}}_{d1}, \acute{\mathbf{C}}_{d2}, \ldots, \acute{\mathbf{C}}_{d\,ndblk}\right] \\
\acute{\mathbf{D}}_d &= \begin{bmatrix} \acute{\mathbf{D}}_{d1} \\ \acute{\mathbf{D}}_{d2} \\ \vdots \\ \acute{\mathbf{D}}_{d\,ndblk} \end{bmatrix}
\end{aligned}
$$

$$
\acute{\mathbf{E}}_d = \begin{bmatrix}
\acute{\mathbf{B}}_{d1} e_d(1,1) & \acute{\mathbf{B}}_{d1} e_d(1,2) & \cdots & \acute{\mathbf{B}}_{d1} e_d(1, ncblk) \\
\acute{\mathbf{B}}_{d2} e_d(2,1) & \acute{\mathbf{B}}_{d2} e_d(2,2) & & \\
\vdots & \vdots & \ddots & \\
\acute{\mathbf{B}}_{d\,md} e_d(md,1) & & & \acute{\mathbf{B}}_{d\,md} e_d(md, ncblk)
\end{bmatrix}
$$

where $e_d(i,j)$ is equal to 1 if block $\mathbf{C}_\mathrm{j}$ is an input to block $\mathbf{D}_\mathrm{i}$, otherwise it is equal to zero.

$$
\acute{\mathbf{F}}_d = \begin{bmatrix}
\acute{\mathbf{D}}_{d1} f_d(1,1) & \acute{\mathbf{D}}_{d1} f_d(1,2) & \cdots & \acute{\mathbf{D}}_{d1} f_d(1, ncblk) \\
\acute{\mathbf{D}}_{d2} f_d(2,1) & \acute{\mathbf{D}}_{d2} f_d(2,2) & & \\
\vdots & \vdots & \ddots & \\
\acute{\mathbf{D}}_{d\,ndblk} f_d(ndblk,1) & & & \acute{\mathbf{D}}_{d\,ndblk} f_d(ndblk, ncblk)
\end{bmatrix}
$$

where $f_d(i,j)$ is equal to 1 if block $\mathbf{C}_\mathrm{j}$ is an input to block $\mathbf{D}_\mathrm{i}$, otherwise it is equal to zero.

$$\acute{\mathbf{G}}_d = \begin{bmatrix} \acute{\mathbf{B}}_{d1} g_d(1,1) & \acute{\mathbf{B}}_{d1} g_d(1,2) & \cdots & \acute{\mathbf{B}}_{d1} g_d(1, nshblk) \\ \acute{\mathbf{B}}_{d2} g_d(2,1) & \acute{\mathbf{B}}_{d2} g_d(2,2) & & \\ \vdots & \vdots & \ddots & \\ \acute{\mathbf{B}}_{d\,md} g_d(md,1) & & & \acute{\mathbf{B}}_{d\,md} g_d(md, nshblk) \end{bmatrix}$$

where $g_d(i,j)$ is equal to 1 if block $\mathbf{S}_j$ is an input to block $\mathbf{D}_i$, otherwise it is equal to zero.

$$\acute{\mathbf{H}}_d = \begin{bmatrix} \acute{\mathbf{D}}_{d1} h_d(1,1) & \acute{\mathbf{D}}_{d1} h_d(1,2) & \cdots & \acute{\mathbf{D}}_{d1} h_d(1, nshblk) \\ \acute{\mathbf{D}}_{d2} h_d(2,1) & \acute{\mathbf{D}}_{d2} h_d(2,2) & & \\ \vdots & \vdots & \ddots & \\ \acute{\mathbf{D}}_{d\,ndblk} h_d(ndblk,1) & & & \acute{\mathbf{D}}_{d\,ndblk} h_d(ndblk, nshblk) \end{bmatrix}$$

where $h_d(i,j)$ is equal to 1 if block $\mathbf{S}_j$ is an input to block $\mathbf{D}_i$, otherwise it is equal to zero.

$$\mathbf{P}_d = \begin{bmatrix} \acute{\mathbf{B}}_{d1} p_d(1,1) & \acute{\mathbf{B}}_{d1} p_d(1,2) & \cdots & \acute{\mathbf{B}}_{d1} p_d(1, ndblk) \\ \acute{\mathbf{B}}_{d2} p_d(2,1) & \acute{\mathbf{B}}_{d2} p_d(2,2) & & \\ \vdots & \vdots & \ddots & \\ \acute{\mathbf{B}}_{d\,md} p_d(md,1) & & & \acute{\mathbf{B}}_{d\,md} p_d(md, ndblk) \end{bmatrix}$$

where $p_d(i,j)$ is equal to 1 if block $\mathbf{D}_j$ is an input to block $\mathbf{D}_i$, otherwise it is equal to zero.

$$\mathbf{Q}_d = \begin{bmatrix} \acute{\mathbf{D}}_{d1} q_d(1,1) & \acute{\mathbf{D}}_{d1} q_d(1,2) & \cdots & \acute{\mathbf{D}}_{d1} q_d(1, ndblk) \\ \mathbf{D}_{d2} q_d(2,1) & \mathbf{D}_{d2} q_d(2,2) & & \\ \vdots & \vdots & \ddots & \\ \acute{\mathbf{D}}_{d\,ndblk} q_d(ndblk,1) & & & \acute{\mathbf{D}}_{d\,ndblk} q_d(ndblk, ndblk) \end{bmatrix}$$

where $q_d(i,\ j)$ is equal to 1 if block $\mathbf{D}_j$ is an input to block $\mathbf{D}_i$, otherwise it is equal to zero.

Solving for the vector $\mathbf{y}_d$, the system can be written as

$$\begin{aligned} \dot{\mathbf{x}}_d(t_{k+1}) &= \mathbf{A}_d \mathbf{x}_d(t_k) + \mathbf{B}_d u_d(t_k) + \mathbf{E}_d \mathbf{x}_c(t_k) + \mathbf{G}_d \mathbf{x}_s(t_k) \\ \mathbf{y}_d(t_k) &= \mathbf{C}_d \mathbf{x}_d(t_k) + \mathbf{D}_d u_d(t_k) + \mathbf{F}_d \mathbf{x}_c(t_k) + \mathbf{H}_d \mathbf{x}_s(t_k) \end{aligned} \tag{7.12}$$

where

$$\begin{aligned} \mathbf{C}_d &= [\mathbf{I} - \mathbf{Q}]^{-1}\, \acute{\mathbf{C}}_d \\ \mathbf{D}_d &= [\mathbf{I} - \mathbf{Q}]^{-1}\, \acute{\mathbf{D}}_d \\ \mathbf{F}_d &= [\mathbf{I} - \mathbf{Q}]^{-1}\, \acute{\mathbf{F}}_d \mathbf{C}_c \\ \mathbf{H}_d &= [\mathbf{I} - \mathbf{Q}]^{-1}\, \{\acute{\mathbf{F}}_d \mathbf{D}_c + \acute{\mathbf{H}}_d\} \\ \mathbf{A}_d &= \acute{\mathbf{A}}_d + \mathbf{P}_d \mathbf{C}_d \\ \mathbf{B}_d &= \acute{\mathbf{B}}_d + \mathbf{P}_d \mathbf{D}_d \\ \mathbf{E}_d &= \acute{\mathbf{E}}_d \mathbf{C}_c + \mathbf{P}_d \mathbf{F}_d \\ \mathbf{G}_d &= \acute{\mathbf{E}}_d \mathbf{D}_c + \acute{\mathbf{G}}_d + \mathbf{P}_d \mathbf{H}_d \end{aligned}$$

Like the continuous case, the nondynamic $\mathbf{D}_i$ blocks will not affect the dimension of the system matrix in (7.12).

For the sample-hold part of the system, the vector $\mathbf{x}_s$, whose elements represents the output of the S_i blocks, will have the following form,

$$\mathbf{x}_s(t_{k+1}) = \mathbf{B}_{sc}\mathbf{x}_c(t_k) + \mathbf{B}_{sd}\mathbf{x}_d(t_k) + \mathbf{B}_{ss}\mathbf{x}_s(t_k) + \mathbf{B}_{su}u_d(t_k) \qquad (7.13)$$

Note that no input u_s to a S_i block is defined and that u_d is a scalar connected to a discrete block. Only a single u_d input is allowed since transfer functions will only be defined between two $\mathbf{D}_i$ blocks. Note also that $\mathbf{B}_{sc}$ is the contribution from the continuous dynamics even though $\mathbf{C}_i$ blocks are not connected to $\mathbf{S}_i$ blocks. This contribution occurs when a $\mathbf{D}_i$ block provides a direct connection between a $\mathbf{C}_i$ and $\mathbf{S}_i$ block, i.e., when the z plane transfer function associated with $\mathbf{D}_i$ has equal number (including none) of poles and zeros .

Before defining the matrices $\mathbf{B}_{sc}$, $\mathbf{B}_{sd}$, $\mathbf{B}_{ss}$, and $\mathbf{B}_{su}$, constraints on the inputs to block $\mathbf{S}_i$ are presented. At the present time, the implementation of LCAP2 only allows one input for each $\mathbf{S}_i$ block, either from a discrete block $\mathbf{D}_j$ with the same sampling period as block $\mathbf{S}_i$ or from another sample-hold block $\mathbf{S}_l$ with a different sampling period than block $\mathbf{S}_i$. In modeling a connection between discrete block $\mathbf{D}_j$ and sample-hold block $\mathbf{S}_i$ which have different sampling periods, (1) define a new sample-hold block $\mathbf{S}_l$ with a sampling period equal to discrete block $\mathbf{D}_j$ and (2) insert block $\mathbf{S}_l$ between $\mathbf{D}_j$ and $\mathbf{S}_i$. Matrices $\mathbf{B}_{sc}$, $\mathbf{B}_{sd}$, $\mathbf{B}_{ss}$, and $\mathbf{B}_{su}$ are then defined as follows:

- If block $\mathbf{S}_i$ is not sampled at t_{k+1}, all elements of row i of $\mathbf{B}_{sc}$, $\mathbf{B}_{sd}$, $\mathbf{B}_{ss}$, and $\mathbf{B}_{su}$ are zero except for column i of $\mathbf{B}_{ss}$ which is equal to one. This nonzero element allows the i-th state of $\mathbf{x}_s$ to be propagated unchanged between t_k and t_{k+1}.
- If block $\mathbf{S}_i$ is sampled at t_{k+1} and the input to block $\mathbf{S}_i$ is from discrete block $\mathbf{D}_j$ which has the same sampling period as $\mathbf{S}_i$, the i-th row of $\mathbf{B}_{sc}$, $\mathbf{B}_{sd}$, $\mathbf{B}_{ss}$, and $\mathbf{B}_{su}$ will be equal to the the j-th row of $\mathbf{F}_d$, $\mathbf{C}_d$, $\mathbf{H}_d$, $\mathbf{D}_d$, respectively.
- If block $\mathbf{S}_i$ is sampled at t_{k+1} and the input to block $\mathbf{S}_i$ is from block $\mathbf{S}_l$, all elements of row i of $\mathbf{B}_{sc}$, $\mathbf{B}_{sd}$, $\mathbf{B}_{ss}$, and $\mathbf{B}_{su}$ are zero except for column l of $\mathbf{B}_{ss}$ which is equal to one.

The second part to computing the state space representation for a continuous-discrete system is the use of the Kalman-Bertram method to compute the state transition matrix from t=0 to t=T, where T is the LCM sampling period of the $\mathbf{D}_i$ and $\mathbf{S}_i$ blocks. From this state transition matrix, eigenvalues, transfer function, root locus, frequency response, and time response can be computed.

The following discussion outlines the steps used in implementing the Kalman-Bertram method of analysis. For details on this method, see Ref. 3.

Step 1 From all the $\mathbf{D}_i$ and $\mathbf{S}_i$ connection blocks, compute the LCM sampling period and set up discrete time segment tables. The discrete time segments correspond to the times at which any $\mathbf{D}_i$ or $\mathbf{S}_i$ block is to be sampled. All samplers are assumed to be synchronized at time zero. This set of discrete times is described by the following sequence,

$$t_0, t_1, t_2, \ldots, t_n$$

where, $t_0 = 0$ and $t_n = $ T.

Step 2 The state vector is written as

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_c \\ \mathbf{x}_d \\ \mathbf{x}_s \end{bmatrix} \tag{7.14}$$

Step 3 The solution of the continuous equations between discrete sampling times t_k and t_{k+1} is given by

$$\mathbf{x}_c(t_k + \tau) = \mathbf{\Phi}_c(\tau)\mathbf{x}_c(t_k) + \int_0^{\tau} \mathbf{\Phi}_c(\tau)\mathbf{B}_c d\tau \; \mathbf{x}_s(t_k) \tag{7.15}$$

or in state space form as

$$\begin{bmatrix} \mathbf{x}_c(t_k + \tau) \\ \mathbf{x}_d(t_k + \tau) \\ \mathbf{x}_s(t_k + \tau) \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi}_c(\tau) & 0 & \int_0^{\tau} \mathbf{\Phi}_c(\tau)\mathbf{B}_c d\tau \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{x}_c(t_k) \\ \mathbf{x}_d(t_k) \\ \mathbf{x}_s(t_k) \end{bmatrix} \tag{7.16}$$

where, $\tau \leq (\tau_{k+1} - \tau_k)$ and $\mathbf{\Phi}_c(\tau) = \mathrm{e}^{\mathbf{A}_c\tau}$.

The state transition matrix propagating the continuous dynamics from t_k to t_{k+1} is given by

$$\begin{bmatrix} \mathbf{x}_c(t_{k+1}) \\ \mathbf{x}_d(t_{k+1}) \\ \mathbf{x}_s(t_{k+1}) \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi}_c(\tau) & 0 & \int_0^{\tau} \mathbf{\Phi}_c(\tau)\mathbf{B}_c d\tau \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{x}_c(t_k) \\ \mathbf{x}_d(t_k) \\ \mathbf{x}_s(t_k) \end{bmatrix} \tag{7.17}$$

where $\tau = t_{k+1} - t_k$. This can then be rewritten as

$$\mathbf{x}(t_{k+1}) = \mathbf{\Phi}_{k+1}\mathbf{x}(t_k) \tag{7.18}$$

where $\mathbf{\Phi}_{k+1}$, the continuous transition matrix, is defined as

$$\mathbf{\Phi}_{k+1} = \begin{bmatrix} \mathbf{\Phi}_c(\tau) & 0 & \int_0^{\tau} \mathbf{\Phi}_c(\tau)\mathbf{B}_c d\tau \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \Big|_{\tau = t_{k+1} - t_k} \tag{7.19}$$

Step 4 The state transition matrix for the discrete equations is given by

$$\begin{bmatrix} \mathbf{x}_c(t_{k+1}) \\ \mathbf{x}_d(t_{k+1}) \\ \mathbf{x}_s(t_{k+1}) \end{bmatrix} = \begin{bmatrix} I & 0 & 0 \\ \mathbf{E}_d & \mathbf{A}_d & \mathbf{G}_d \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{x}_c(t_k) \\ \mathbf{x}_d(t_k) \\ \mathbf{x}_s(t_k) \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{B}_d \\ 0 \end{bmatrix} u_d(t_k) \tag{7.20}$$

which can be rewritten as

$$\mathbf{x}(t_{k+1}) = \mathbf{D}_{k+1}\mathbf{x}(t_k) + \begin{bmatrix} 0 \\ \mathbf{B}_d \\ 0 \end{bmatrix} u_d(t_k) \tag{7.21}$$

where $\mathbf{D}_{k+1}$[1] , the discrete transition matrix, is given by

$$\mathbf{D}_{k+1} = \begin{bmatrix} I & 0 & 0 \\ \mathbf{E}_d & \mathbf{A}_d & \mathbf{G}_d \\ 0 & 0 & I \end{bmatrix} \tag{7.22}$$

[1] The bold face $\mathbf{D}_k$ has two different meanings in this report (the other is the k-th discrete connection block). The context in which it is used will not present any ambiguity.

Step 5 The state transition matrix for the sample-hold equations is given by

$$\begin{bmatrix} \mathbf{x}_c(t_{k+1}) \\ \mathbf{x}_d(t_{k+1}) \\ \mathbf{x}_s(t_{k+1}) \end{bmatrix} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ \mathbf{B}_{sc} & \mathbf{B}_{sd} & \mathbf{B}_{ss} \end{bmatrix} \begin{bmatrix} \mathbf{x}_c(t_k) \\ \mathbf{x}_d(t_k) \\ \mathbf{x}_s(t_k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \mathbf{B}_{su} \end{bmatrix} u_d(t_k) \tag{7.23}$$

which can be rewritten as

$$\mathbf{x}(t_{k+1}) = \mathbf{S}_{k+1}\mathbf{x}(t_k) + \begin{bmatrix} 0 \\ 0 \\ \mathbf{B}_{su} \end{bmatrix} u_d(t_k) \tag{7.24}$$

where $\mathbf{S}_{k+1}$[1] , the sample-hold transition matrix, is given by

$$\mathbf{S}_{k+1} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ \mathbf{B}_{sc} & \mathbf{B}_{sd} & \mathbf{B}_{ss} \end{bmatrix} \tag{7.25}$$

Step 6 Compute the state transition matrix of the system from t=0 to t=T. This is obtained by computing the state transition matrix at each discrete time and then propagating the results from t=0 to t=T.

First consider the case with input $u_d = 0$. For the k-th discrete time, the state transition matrix from t_k and t_{k+1} is given by the following sequence,

$$\mathbf{x}({t_k}^+) = \mathbf{S}_{k+1}\mathbf{x}(t_k) \tag{7.26}$$

$$\mathbf{x}({t_k}^{++}) = \mathbf{D}_{k+1}\mathbf{x}({t_k}^+) = \mathbf{D}_{k+1}\mathbf{S}_{k+1}\mathbf{x}(t_k) \tag{7.27}$$

$$\mathbf{x}(t_{k+1}) = \mathbf{\Phi}_{k+1}\mathbf{x}({t_k}^{++}) = \mathbf{\Phi}_{k+1}\mathbf{D}_{k+1}\mathbf{S}_{k+1}\mathbf{x}(t_k) \tag{7.28}$$

where, ${t_k}^+$ is defined to be slightly larger than t_k and ${t_k}^{++}$ is defined to be slightly larger than ${t_k}^+$.

Defining Ψ_k as

$$\mathbf{\Psi}_k = \mathbf{\Phi}_k\mathbf{D}_k\mathbf{S}_k \tag{7.29}$$

the state transition matrix from t_k to t_{k+1} can be written as

$$\mathbf{x}(t_{k+1}) = \mathbf{\Psi}_{k+1}\mathbf{x}(t_k) \tag{7.30}$$

Thus, the state transition matrix between t_0 and t_n is given by

$$\mathbf{x}(t_n) = \mathbf{\Psi}_n\mathbf{\Psi}_{n-1}\ldots\mathbf{\Psi}_2\mathbf{\Psi}_1\mathbf{x}(t_0) \tag{7.31}$$

Defining Ψ as

$$\mathbf{\Psi} = \mathbf{\Psi}_n\mathbf{\Psi}_{n-1}\ldots\mathbf{\Psi}_2\mathbf{\Psi}_1 \tag{7.32}$$

Eq. (7.30) can be rewritten as

$$\mathbf{x}(t_n) = \mathbf{\Psi}\mathbf{x}(t_0) \tag{7.33}$$

[1] The bold face $\mathbf{S}_k$ has two different meanings in this report (the other is the k-th sample-hold connection block). The context in which it is used will not present any ambiguity.

which after transforming to the z plane, yields

$$(z\mathbf{I} - \Psi)\mathbf{x}(z) = 0 \tag{7.34}$$

If Ψ is the state transition matrix for a closed loop system, its eigenvalues are the closed loop poles. This operation can be implemented by the command B2EIG. Command B2LOCI will compute the loci of eigenvalues as a function of gain applied to any $\mathbf{C}_i$ or $\mathbf{D}_i$ block. These two commands are described in the Reference in Appendix A.

Now the case where the input u_d is not zero will be considered. Since a transfer function between two $\mathbf{D}_i$ blocks are to be computed at the LCM sampling period, the input u_d is only applied at t = 0, T, 2T, 3T, ...

The state transition matrix between t_0 and t_1 is computed by the following sequence

$$\mathbf{x}(t_0{}^{+}) = \mathbf{S}_1\mathbf{x}(t_0) + \begin{bmatrix} 0 \\ 0 \\ \mathbf{B}_{su} \end{bmatrix} u_d(t_0) \tag{7.35}$$

$$\begin{aligned} \mathbf{x}(t_0{}^{++}) &= \mathbf{D}_1\mathbf{x}(t_0{}^{+}) + \begin{bmatrix} 0 \\ \mathbf{B}_d \\ 0 \end{bmatrix} u_d(t_0) \\ &= \mathbf{D}_1\mathbf{S}_1\mathbf{x}(t_0) + \left\{ \mathbf{D}_1 \begin{bmatrix} 0 \\ 0 \\ \mathbf{B}_{su} \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{B}_d \\ 0 \end{bmatrix} \right\} u_d(t_0) \end{aligned} \tag{7.36}$$

$$\begin{aligned} \mathbf{x}(t_1) &= \Phi_1\mathbf{x}(t_0{}^{++}) \\ &= \Phi_1\mathbf{D}_1\mathbf{S}_1\mathbf{x}(t_0) + \Phi_1 \left\{ \mathbf{D}_1 \begin{bmatrix} 0 \\ 0 \\ \mathbf{B}_{su} \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{B}_d \\ 0 \end{bmatrix} \right\} u_d(t_0) \end{aligned} \tag{7.37}$$

Defining $\mathbf{B}_1$ as

$$\mathbf{B}_1 = \Phi_1 \left\{ \mathbf{D}_1 \begin{bmatrix} 0 \\ 0 \\ \mathbf{B}_{su} \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{B}_d \\ 0 \end{bmatrix} \right\} \tag{7.38}$$

Eq. (7.37) can be simplified as

$$\mathbf{x}(t_1) = \Psi_1\mathbf{x}(t_0) + \mathbf{B}_1 u_d(t_0) \tag{7.39}$$

Since u_d will be zero until t_n, the transition matrix from t_0 to t_n is

$$\mathbf{x}(t_n) = \Psi_n\Psi_{n-1}\cdots\Psi_2\Psi_1\mathbf{x}(t_0) + \Psi_n\Psi_{n-1}\cdots\Psi_2\mathbf{B}_1 u_d(t_0) \tag{7.40}$$

Using (7.31) and defining $\mathbf{B}$ as

$$\mathbf{B} = \Psi_n\Psi_{n-1}\cdots\Psi_2\mathbf{B}_1 \tag{7.41}$$

Eq. (7.40) can be rewritten as

$$\mathbf{x}(t_n) = \Psi\mathbf{x}(t_0) + \mathbf{B}u_d(t_0) \tag{7.42}$$

The state space representation of the discrete system at the LCM sampling rate, T, is thus

$$\begin{aligned} \mathbf{x}(kT+T) &= \mathbf{A}\mathbf{x}(kT) + \mathbf{B}u_d(kT) \\ \mathbf{y}_d(kT) &= \mathbf{C}\mathbf{x}(kT) + \mathbf{D}u_d(kT) \end{aligned} \tag{7.43}$$

where

$$\begin{aligned} \mathbf{A} &= \Psi \\ \mathbf{C} &= [\mathbf{F}_d \ \mathbf{C}_d \ \mathbf{H}_d] \\ \mathbf{D} &= \mathbf{D}_d \end{aligned}$$

There are two different methods for evaluating a SISO transfer function from (7.43). The first method, which is implemented by command B2TF, computes the poles and zeros of the z plane transfer function from block $\mathbf{D}_i$ to $\mathbf{D}_j$ using the following equation

$$\frac{y_{dj}(z)}{u(z)} = \frac{\det \begin{bmatrix} z\mathbf{I} - \mathbf{A} & \mathbf{B} \\ \text{row j of } -\mathbf{C} & \text{row j of } \mathbf{D} \end{bmatrix}}{\det \begin{bmatrix} z\mathbf{I} - \mathbf{A} \end{bmatrix}} \tag{7.44}$$

Like the command B1TF, the QZ method, which is used to compute the zeros, is sensitive to both (1) the order of the $\mathbf{SPTF}_i$ and $\mathbf{ZPTF}_i$ blocks associated with the $\mathbf{C}_i$ and $\mathbf{D}_i$ blocks, respectively and (2) the method used to convert from transfer function to state space representation. Future effort to address this problem was given in Section 7.2.1. This sensitivity of the QZ method will be more pronounced for discrete systems since the roots are in the z plane.

The second method for evaluating an SISO transfer function from (7.43) is implemented by the command B2FREQ which uses Laub's state space method. The transfer from block $\mathbf{C}_i$ to block $\mathbf{C}_j$ is give⁻ by the j-th row of

$$\frac{\mathbf{Y}(z)}{u(z)} = \mathbf{C}[\, z\mathbf{I} - \mathbf{A}\,]^{-1}\mathbf{B} + \mathbf{D} \tag{7.45}$$

The comments on the command B1FREQ g ven in Section 7.2.1 apply to command B2FREQ as well. Since computation of poles and zeros of transfer function is a difficult numerical problem, particularly in the z plane, the analyst is encouraged to use the command B2FREQ to check the results of command B2TF if there are any doubts on the accuracy of the computed poles and zeros.

For small order problems, a time response can be computed by first evaluating a transfer function in rational form by using the command B2TF and then using command ZTIME which will compute the time response by recursive evaluation of a difference equation. For higher order systems, this method will not work since the z plane coefficients cannot be accurately represented with enough precision by the computer.

Command B2TIME computes the the SISO time response from block $\mathbf{D}_i$ to block $\mathbf{D}_j$ by using the solution of (7.43) which is

$$\begin{aligned} \mathbf{x}(kT+T) &= \mathbf{A}\mathbf{x}(kT) + \mathbf{B}u(kT) \\ \text{row j of } [\mathbf{y}(kT) &= \mathbf{C}\mathbf{x}(kT) + \mathbf{D}u(kT)] \end{aligned} \tag{7.46}$$

7.3 LCAP2 Commands for Automated Analysis

After a continuous or a continuous-discrete multirate system is modeled as a connection of transfer function blocks, the following types of commands are available for automated analysis:

- Eigenvalue
- Transfer function evaluation
- Root locus
- Frequency response calculation using Laub's method
- Time response using state space method

Before any of the these commands can be used, (1) a system must be properly connected as described in Section 7.1 and (2) have all the transfer functions that are to be connected stored in the appropriate $\mathbf{SPTF}_i$ and $\mathbf{ZPTF}_i$ transfer functions. Connection of transfer function blocks can be done before or after $\mathbf{SPTF}_i$ and $\mathbf{ZPTF}_i$ data is loaded or computed.

Along with the above commands, there are commands for saving and loading transfer function connection data. The use of all the above types of commands will be demonstrated in the next two subsections for the examples in Figures 7.1 and 7.2.

7.3.1 Continuous Systems

Continuing the example in Section 7.1.1, code will be presented to demonstrate the commands that are available for automated analysis of continuous systems modeled as a connection of transfer functions.

The following sequence of operations is to be performed:

- Compute closed loop poles using the B1EIG eigenvalue command.
- Compute closed loop transfer function between the input u and the output of $\mathbf{C}_3$.
- Evaluate closed loop frequency response between the input u and the output of $\mathbf{C}_3$ using Laub's method.
- Evaluate time response between input u and the output of $\mathbf{C}_3$ using state space method.
- Evaluate time response between input u and the output of C_3 using inverse Laplace transform method.
- Save connection data of closed loop configuration.

- Compute open loop transfer function of the inner loop.
- Compute root locus by varying the forward loop gain.

The FORTRAN code to implement these operations can be written as:

```
      CALL B1EIG(5)                      "eigenvalues stored in POLY5"
C
      UCIN=1                             "input u connected to C1"
      UMAGN=1.                           "magnitude of input"
      YCOUT=3                            "output of transfer function is C3"
      CALL B1TF(8)                       "computed transfer function stored in SPTF8"
C
      NOMEGA=2                           "number of frequencies in array OMEGA"
      OMEGA(1)=1                         "see Reference on B1FREQ"
      OMEGA(2)=10                        "see Reference on B1FREQ"
      UCIN=1                             "input u connected to C1"
      YCOUT=3                            "output of transfer function is C3"
      CALL B1FREQ( )                     "evaluate frequency response without computing
C                                         the poles and zeros"
      TDELT=.1                           "delta time for time response"
      TEND=5.                            "end time for time response"
      CALL B1TIME ()                     "time response by state space method"
      TZERO=0.                           "start time for time response (required by STIME)"
      CALL STIME(8)                      "time respones by inverse Laplace transform method"
      CALL B1SAVE('CLSLP1')              "save connection data to file CLSLP1"
C
      CALL B1INIT('OPEN INNER LOOP CONFIG.','OLD',5)    "initialize with old data"
      IYCIN(1)=-5                        "connect only outer loop"
      CALL B1CEQ('SUMMER BLOCK',1,0,1,IYCIN)
      CALL B1END                         "inner loop is now opened"
C
      UCIN=2                             "input u connected to C2"
      YCOUT=1                            "output of transfer function is C1"
      CALL B1TF(9)                       "computed transfer function stored in SPTF9"
C
      CALL B1LOAD('CLSLP1')              "reload closed loop configuration"
C
      NLOCI=2                            "number of root locus gains in array [illegible]"
      KGAIN(1)=2                         "see Reference on B1LOCI"
      KGAIN(2)=20                        "see Reference on B1LOCI"
      KFLG=0                             "see Reference on B1LOCI"
      KDELT=2                            "see Reference on B1LOCI"
      CALL B1LOCI(2)                     "vary gain of C2 block"
```

For a detail description of the B1EIG, B1TF, B1FREQ, B1TIME, B1LOCI, B1SAVE, and B1LOAD commands, see the Reference in Appendix A.

7.3.2 Continuous-Discrete Multirate Systems

Continuing the example in Section 7.1.2, code will be presented to demonstrate the commands that are available for automated analysis of continuous systems modeled as a connection of transfer functions.

The following sequence of operations is to be performed:

- Compute closed loop poles using the B2EIG eigenvalue command.
- Compute closed loop transfer function between the input R and the output of $\mathbf{D}_7$.
- Evaluate closed loop frequency response using Laub's method.
- Evaluate closed loop time response using state space method.
- Save connection data of closed loop configuration.
- Open inner loop at point P_1 and compute open loop transfer function between input to block $\mathbf{D}_6$ and the output of block $\mathbf{D}_2$.
- Reload connection data for closed loop configuration
- Compute root locus by varying the forward loop gain.

The FORTRAN code to implement these operations can be written as:

```
      CALL B2EIG(6)                   "eigenvalues stored in POLY6"
C
      UDIN=1                          "input u connected to Di"
      UMAGN=1.                        "magnitude of input"
      YDOUT=7                         "output of transfer function is D7"
      CALL B2TF(8)                    "computed transfer function stored in ZPTF8"
C
      NOMEGA=2                        "number of frequencies in array OMEGA"
      OMEGA(1)=1                      "see Reference on B2FREQ"
      OMEGA(2)=10                     "see Reference on B2FREQ"
      UDIN=1                          "input u connected to D1"
      YDOUT=7                         "output of transfer function is D7"
      CALL B2FREQ( )                  "evaluate frequency response without computing
C                                      the poles and zeros"
      SAMPT=.02                       "sampling period for evaluating time response"
      TEND=4.                         "end time for time response"
      CALL B2TIME( )                  "evaluate time response using state space method"
      CALL B2SAVE('CLSLP2')           "save connection data to file CLSLP2"
C
      CALL B2INIT('OPEN INNER LOOP CONFIG.','OLD',5)   "initialize with old data"
      IYCIN(1)=-1                     "connection for block D1"
      IYCIN(2)=-3                     "connection for outer loop:
      CALL B2DEQ('SUMMER BLOCK',6,0,T1,2,IYDIN)
      CALL B2END                    “inner loop is now opened”
C
      UDIN=6                        “input u connected to $\mathbf{D}_6$”
      YDOUT=2                       “output of transfer function is $\mathbf{D}_2$”
      CALL B2TF(9)                  “computed transfer function stored in $\mathbf{ZPTF}_9$
C
      CALL B2LOAD('CLSLP2')         “reload closed loop configuration”
C
      NLOCI=2                       “number of root locus gains in array KGAIN”
      KGAIN(1)=2                    “see Reference on B1LOCI”
      KGAIN(2)=20                   “see Reference on B1LOCI”
      KFLG=0                        “see Reference on B1LOCI”
      KDELT=2                       “see Reference on B1LOCI”
      CALL B2LOCI('D',2)            “vary gain of $\mathbf{D}_2$ block”
```

Note that the closed loop frequency response could also be computed by the command ZFREQ after the closed loop transfer function was computed and stored in $\mathbf{ZPTF}_8$. For a detail description of the B2EIG, B2TF, B2FREQ, B2TIME, B2LOCI, B2SAVE, and B2LOAD commands, see the Reference in Appendix A.

Chapter 8

Basic Examples

A selection of basic LCAP2 commands are presented in this chapter. Examples 1 through 12 were prepared to be be executed sequentially in one batch job. Example 12 demonstrates the use of the SAVE command used to save transfer function, polynomial, and matrix data for a restart capability. Example 13 demonstrates the LOAD command used to restore the data saved in Example 12.

Each example begins with a statement of the problem and is followed by two types of user input code, one being all FORTRAN code[1] and the other a combination of FORTRAN and PRECMP code[2] which requires fewer input statements. After the input code, the printer output is given.

Examples 1 through 12 can be reproduced by the users by executing any of the following files which are saved as indirect files on CDC MFB. A copy of an indirect file can be obtained by typing the following command in INTERCOM:

```
IGET,file_name/PF=ZL2USER,ID=9487
```

where `file_name` and its description is given below in Table 8.1.

Table 8.1: Decks for Creation of Examples 1-12

Files For Reproducing Chapter 8 Examples	
file_name	Description
CBUSER	For CRAY Without PRECMP
CPBUSER	For CRAY With PRECMP
BBUSER	For CDC MFB Without PRECMP
BPBUSER	For CDC MFB With PRECMP
XBUSER	For CDC MFX Without PRECMP
XPBUSER	For CDC MFX With PRFCMP

To execute a CRAY job, type:

```
CSUBMIT,file_name.
```

[1] Created by files CBUSER, BBUSER, or XBUSER in Table 8.1.

[2] Created by files CPBUSER, BPBUSFR, or XPBUSER in Table 8.1

To execute a CDC job, type:

```
BATCH,file_name,INPUT.
```

In Examples 1,5,6,7,9, and 10, which produce both low and high resolution plots, only the printer plots are shown. The high resolution hardcopy plot files generated by these examples are processed by the HARDCPY program after the LCAP2 program has been executed. The hardcopy plots from these examples are presented in Appendix H.

8.1 Example 1 - S Plane Frequency Response

Problem: Compute frequency response of

$$\frac{a^2}{s^2 + 2\zeta as + a^2}$$

between 0.10 to 100. rad/sec, where $a = 5.$ and $\zeta = .5$

Data will be loaded in coefficient form using the command SPLDC. Polynomial arrays POLYN and POLYD are used with SPLDC.

The FORTRAN code for this example is:

```
C       EXAMPLE 1
C       LOAD IN DATA USING LCAP2 OPERATOR SPLDC AND POLYNOMIAL ARRAYS
C       POLYN AND POLYD
C
        A=5.
        ZETA=.5
        POLYN(1)=0.             “degree of numerator”
        POLYN(2)=A*A            “coefficient of s**0”
        POLYD(1)=2.             “degree of denominator”
        POLYD(2)=A*A            “coefficient of s**0”
        POLYD(3)=2*.ZETA*A      “coefficient of s**1”
        POLYD(4)=1.             “coefficient of s**2”
        CALL SPLDC(1)           “load coefficient data into SPTF1
C       . . . . . . . .
C       ENTER FREQUENCY RESPONSE PARAMETERS FOR USE WITH SFREQ
C
        FAUTO=1                 “.NE.0 (preset=1) for auto. freq. selection mode ”
        RAD=1                   “.NE.0 (preset=1) for rad/sec, otherwise Hz”
        NOMEGA=3                “number of values of OMEGA(i)’s to be entered”
        OMEGA(1)=.1             “OMEGA(1)=first frequency value to be used”
        OMEGA(2)=1.             “user specified frequency value”
        OMEGA(3)=100.           “OMEGA(NOMEGA)=last frequency value”
        FDELAY=0                “time delay (preset=0)”
        FNICO=1                 “.NE.0 (preset=0) for Nichols plot”
        FBODE=1                 “.NE.0 (preset=0) for Bode plots”
```

```
      CYCLE=0                     ".EQ.0 for auto. selection of 2 or 3 cycle for Bode
                                    plots (1 cycle is not available)"
      FNYQS=1                     ".NE.0 (preset=0) for Nyquist plot"
      NQDB=1                      ".NE.0 (preset=0) for hardcopy NYQUIST plot in dB"
      GRAFP=1                     ".NE.0 (preset=1) for low resolution plot"
      HRDCPY=1                    ".NE.0 (preset=0) for high resolution plot"
C     ENTER PLOT TITLE
      TITLE1 = 'EXAMPLE 1 S PLANE FREQUENCY RESPONSE'
      CALL SFREQ(1)               "compute frequency response of SPTF1"
C     . . . . . . .
```

The FORTRAN/PRECMP code for this example is:

```
C     EXAMPLE 1
C     LOAD IN DATA USING LCAP2 OPERATOR SPLDC AND POLYNOMIAL ARRAYS
C     POLYN AND POLYD
C
      A=5.
      ZETA=.5
*POLYN 0 A*A ! DEG. OF NUM., COEFF. IN ASCENDING ORDER
*POLYD 2 A*A 2.*ZETA*A 1 !  DEG. OF DENOM., COEFF. IN ASCENDING ORDER
*SPLDC 1 !  LOAD COEFFICIENT DATA INTO SPTF(1)
C     . . . . . . .
C     ENTER FREQUENCY RESPONSE PARAMETERS FOR USE WITH SFREQ
C
      FAUTO=1                     ".NE.0 (preset=1) for auto. freq. selection mode "
      RAD=1                       ".NE.0 (preset=1) for rad/sec, otherwise Hz"
      NOMEGA=3                    "number of values of OMEGA(i)'s to be entered"
*OMEGA .1 1.  100.  !  OMEGA(1)=FIRST FREQ., OMEGA(NOMEGA)=LAST FREQ.
      FDELAY=0                    "time delay (preset=0)"
      FNICO=1                     ".NE.0 (preset=0) for Nichols plot"
      FBODE=1                     ".NE.0 (preset=0) for Bode plots"
      CYCLE=0                     ".EQ.0 for auto. selection of 2 or 3 cycle for Bode
                                    plots (1 cycle is not available)"
      FNYQS=1                     ".NE.0 (preset=0) for Nyquist plot"
      NQDB=1                      ".NE.0 (preset=0) for hardcopy NYQUIST plot in dB"
      GRAFP=1                     ".NE.0 (preset=1) for low resolution plot"
      HRDCPY=1                    ".NE.0 (preset=0) for high resolution plot"
C     ENTER PLOT TITLE
      TITLE1 = 'EXAMPLE 1 S PLANE FREQUENCY RESPONSE'
*SFREQ 1 !  COMPUTE FREQUENCY RESPONSE OF SPTF(1)
C     . . . . . . .
```

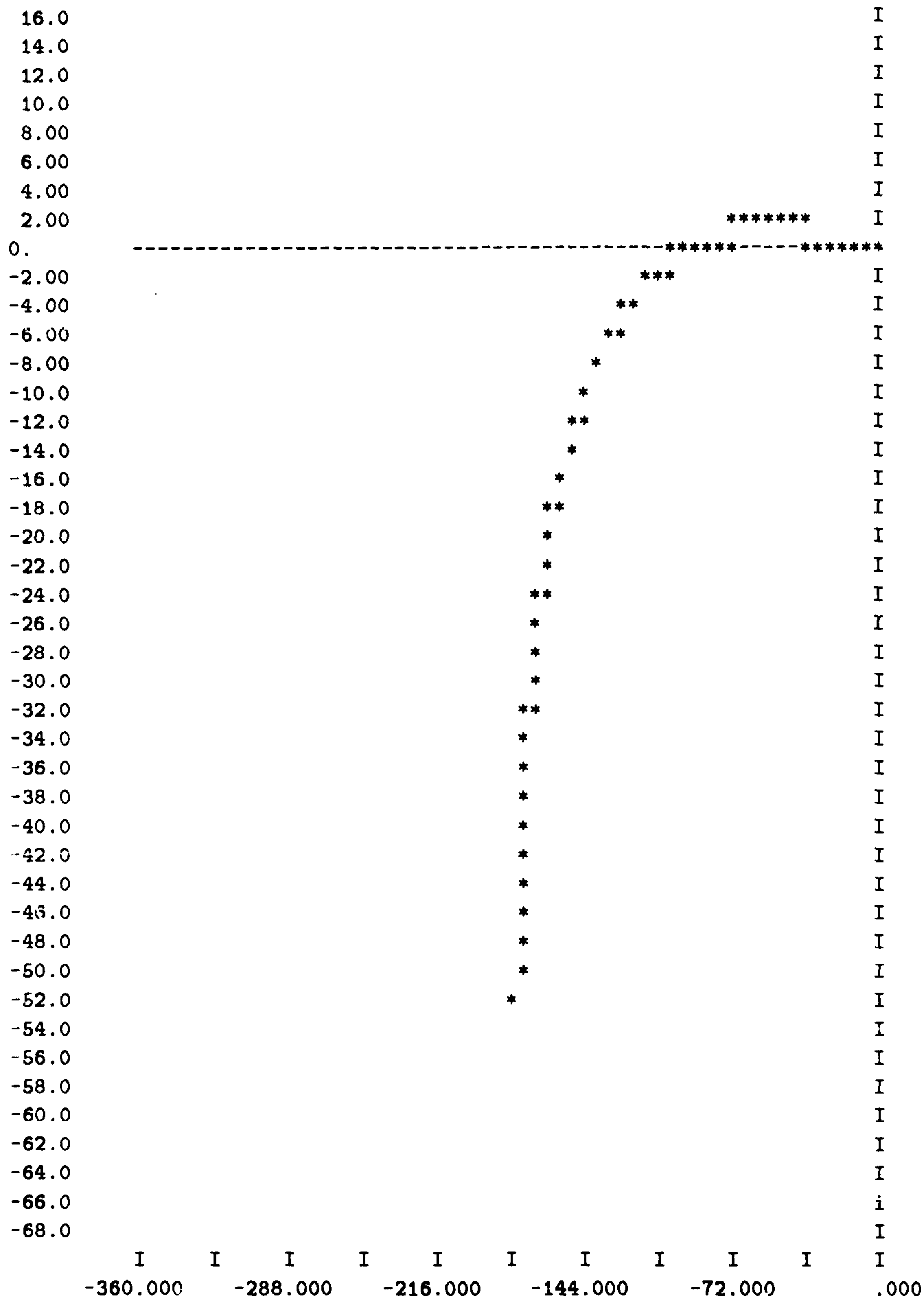
The printer output for this example is:

```
DEGREE OF POLYN   IS  0          (COEFFICIENTS IN ASCENDING ORDER)
25.

DEGREE OF POLYD   IS  2          (COEFFICIENTS IN ASCENDING ORDER)
25. 5. 1.

************************************************************
*    SPLDC - LOAD TRANSFER FUNCTION IN COEFFICIENT FORM    *
************************************************************

DEGREE OF NUMERATOR OF SPTF1   IS  0    (COEFFICIENTS IN ASCENDING ORDER)
25.

DEGREE OF DENOMINATOR OF SPTF1   IS  2  (COEFFICIENTS IN ASCENDING ORDER)
25. 5. 1.
BODE GAIN =    1.0000000
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                               CP=  6.57

************************************************************
*     SFREQ - FREQUENCY RESPONSE OF S-PLANE TRANSFER       *
*             FUNCTION 1                                   *
************************************************************

TRANSPORT DELAY OR DEAD TIME FOR S-PLANE FREQUENCY RESPONSE (FDELAY) =0.

AUTOMATIC FREQUENCY MODE IF FAUTO.NE.O, FAUTO = 1.000

NOMEGA = 3.000    OMEGA = .1000    , 1.000    , 100.0    ,

   OMEGA                 REAL    IMAGINARY      DB     PHASE   PHASE
   RAD/SEC                                                     MARGIN
   .1000              .100E+01  -.200E-01     .002    -1.15   178.85
   .1200              .100E+01  -.240E-01     .003    -1.38   178.62
   .1440              .100E+01  -.288E-01     .004    -1.65   178.35
   .1728              .100E+01  -.346E-01     .005    -1.98   178.02
   .2074              .100E+01  -.415E-01     .007    -2.38   177.62
   .2488              .100E+01  -.499E-01     .011    -2.86   177.14
   .2986              .100E+01  -.599E-01     .015    -3.43   176.57
   .3583              .100E+01  -.720E-01     .022    -4.12   175.88
   .4300              .100E+01  -.866E-01     .032    -4.95   175.05
   .5160              .100E+01  -.104E+00     .046    -5.95   174.05
   .6192              .100E+01  -.126E+00     .066    -7.17   172.83
   .7430              .100E+01  -.152E+00     .095    -8.64   171.36
   .8916              .999E+00  -.184E+00     .136   -10.44   169.56
```

1.000	.998E+00	-.208E+00	.170	-11.77	168.23
1.6	.981E+00	-.406E+00	.520	-22.47	157.53
2.440	.931E+00	-.596E+00	.869	-32.64	147.36
3.080	.812E+00	-.806E+00	1.166	-44.79	135.21
3.560	.657E+00	-.949E+00	1.249	-55.30	124.70
4.040	.449E+00	-.104E+01	1.116	-66.75	113.25
4.520	.215E+00	-.106E+01	.703	-78.57	101.43
5.000	.227E-13	-.100E+01	.000	-90.00	90.00
5.480	-.162E+00	-.883E+00	-.940	-100.40	79.60
6.120	-.285E+00	-.701E+00	-2.421	-112.15	67.85
6.920	-.332E+00	-.503E+00	-4.399	-123.48	56.52
7.880	-.317E+00	-.336E+00	-6.707	-133.27	46.73
8.840	-.278E+00	-.231E+00	-8.834	-140.25	39.75
10.12	-.226E+00	-.148E+00	-11.363	-146.83	33.17
11.40	-.184E+00	-.999E-01	-13.584	-151.50	28.50
13.00	-.144E+00	-.651E-01	-16.014	-155.71	24.29
14.92	-.111E+00	-.418E-01	-18.536	-159.32	20.68
16.84	-.874E-01	-.285E-01	-20.731	-161.96	18.04
19.40	-.661E-01	-.183E-01	-23.275	-164.57	15.43
21.96	-.517E-01	-.124E-01	-25.488	-166.50	13.50
25.16	-.394E-01	-.816E-02	-27.902	-168.31	11.69
29.00	-.297E-01	-.528E-02	-30.410	-169.92	10.08
32.84	-.232E-01	-.361E-02	-32.598	-171.14	8.86
37.96	-.173E-01	-.232E-02	-35.140	-172.37	7.63
43.08	-.135E-01	-.158E-02	-37.354	-173.29	6.71
49.48	-.102E-01	-.104E-02	-39.774	-174.17	5.83
57.16	-.765E-02	-.674E-03	-42.292	-174.96	5.04
64.84	-.595E-02	-.461E-03	-44.489	-175.56	4.44
75.08	-.443E-02	-.297E-03	-47.043	-176.17	3.83
85.32	-.343E-02	-.202E-03	-49.268	-176.63	3.37
98.12	-.260E-02	-.133E-03	-51.700	-177.08	2.92
100.0	-.250E-02	-.125E-03	-52.030	-177.13	2.87

NICHOLS PLOT (MAGN. VS PHASE) 09/07/88

EXAMPLE 1 S PLANE FREQUENCY RESPONSE



```
BODE PLOT (MAGN. VS FREQ.)                                        09/07/88
EXAMPLE 1 S PLANE FREQUENCY RESPONSE

  16.0     I
  14.0     I
  12.0     I
  10.0     I
  8.00     I
  6.00     I
  4.00     I
  2.00     I                         ****
 0.        **-**-**-***-**-**********----**------------------------------
 -2.00     I                               *
 -4.00     I                                *
 -6.00     I                                **
 -8.00     I                                 **
 -10.0     I                                  **
 -12.0     I                                   **
 -14.0     I                                    *
 -16.0     I                                     *
 -18.0     I                                      *
 -20.0     I                                      **
 -22.0     I                                       **
 -24.0     I                                        **
 -26.0     I                                         **
 -28.0     I                                          **
 -30.0     I                                           *
 -32.0     I                                            *
 -34.0     I                                             *
 -36.0     I                                             **
 -38.0     I                                              **
 -40.0     I                                               **
 -42.0     I                                                **
 -44.0     I                                                 **
 -46.0     I                                                  *
 -48.0     I                                                   **
  50.0     I                                                    *
 -52.0     I                                                     *
 -54.0     I
 -56.0     I
 -58.0     I
 -60.0     I
 -62.0     I
 -64.0     I
 -66.0     I
 -68.0     I
           I     I     I     I     I     I     I     I     I     I     I
         .100E+00          1.00              10.0             100.
```

```
BODE PLOT (PHASE VS FREQ.)                                        09/07/88
EXAMPLE 1 S PLANE FREQUENCY RESPONSE

  30.0     I
  20.0     I
  10.0     I
 0.        **-**-**-***--------------------------------------------------
 -10.0     I            ** ****
 -20.0     I                   ****
 -30.0     I                       **
 -40.0     I                        ***
 -50.0     I                          **
 -60.0     I                           **
 -70.0     I                            **
 -80.0     I                             *
 -90.0     I                             **
 -100.     I                              **
 -110.     I                               *
 -120.     I                                *
 -130.     I                                **
 -140.     I                                 ***
 -150.     I                                   ***
 -160.     I                                     ****
 -170.     I                                        **********
 -180.     I                                                 *****
 -190.     I
 -200.     I
 -210.     I
 -220.     I
 -230.     I
 -240.     I
 -250.     I
 -260.     I
 -270.     I
 -280.     I
 -290.     I
 -300.     I
 -310.     I
 -320.     I
 -330.     I
 -340.     I
 -350.     I
 -360.     I
 -370.     I
 -380.     I
 -390.     I
           I     I     I     I     I     I     I     I     I     I     I
         .100E+00          1.00              10.0              100.
```

```
NYQUIST PLOT                                                      09/07/88
EXAMPLE 1 S PLANE FREQUENCY RESPONSE

  4.20                                   I
  4.00                                   I
  3.80                                   I
  3.60                                   I
  3.40                                   I
  3.20                                   I
  3.00                                   I
  2.80                                   I
  2.60                                   I
  2.40                                   I
  2.20                                   I
  2.00                                   I
  1.80                                   I
  1.60                                   I
  1.40                                   I
  1.20                                   I
  1.00                                   I
  .800                                   I
  .600                                   I
  .400                                   I
  .200                                   I
 0.        ----------------------------***-----------*------------------
 -.200                               *** I           *
 -.400                               *   I           *
 -.600                               *   I          *
 -.800                                ** I        ***
 -1.00                                  ***********
 -1.20                                   I
 -1.40                                   I
 -1.60                                   I
 -1.80                                   I
 -2.00                                   I
 -2.20                                   I
 -2.40                                   I
 -2.60                                   I
 -2.80                                   I
 -3.00                                   I
 -3.20                                   I
 -3.40                                   I
 -3.60                                   I
 -3.80                                   I
 -4.00                                   I
 -4.20                                   I
           I     I     I     I     I     I     I     I     I     I     I
        -2.50 -2.00 -1.50 -1.00  -.50   .00   .50  1.00  1.50  2.00  2.50
```

8.2 Example 2 - Roots of a Polynomial

Problem: Compute roots of the polynomial

$$x^3 + 13x^2 + 38s^2 + 34$$

The FORTRAN code for this example is:

```
C       EXAMPLE 2
C       LOAD IN DATA USING LCAP2 OPERATOR PRTS AND POLYNOMIAL ARRAY POLY
C
        POLY(1)=3                  “degree of polynomial”
        POLY(2)=34.                “coefficient of x**0”
        POLY(3)=38.                “coefficient of x**1”
        POLY(4)=13.                “coefficient of x**2”
        POLY(5)=1.                 “coefficient of x**3”
        CALL PLDC(1)               “load coefficient data into POLY1
C       . . . . . . . .
        CALL PRTS(1)               “find roots of POLY1”
C       . . . . . . . .
```

The FORTRAN/PRCMP code for this example is:

```
C       EXAMPLE 2
C       LOAD IN DATA USING LCAP2 OPERATOR PRTS AND POLYNOMIAL ARRAY POLY
C
*POLY 3 34.  38.  13.  1 !  DEG. OF POLY., COEFF. IN ASCENDING ORDER
*PLDC 1 !  LOAD COEFFICIENT DATA INTO POLY(1)
C       . . . . . . . .
*PRTS 1 !  FIND ROOTS OF POLY(1)
C       . . . . . . . .
```

The printer output for this example is:

```
DEGREE OF POLY    IS  3          (COEFFICIENTS IN ASCENDING ORDER)
34. 38. 13. 1.

************************************************************
```

```
*     PLDC - POLYNOMIAL LOAD IN COEFFICIENT FORM             *
**************************************************************

DEGREE OF POLY1    IS  3          (COEFFICIENTS IN ASCENDING ORDER)
34. 38. 13. 1.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                            CP=  6.89
DEGREE OF POLY1    IS  3          (COEFFICIENTS IN ASCENDING ORDER)
34. 38. 13. 1.

**************************************************************
*     PRTS - FIND ROOTS OF POLY1                             *
**************************************************************

                           THE ROOTS OF ROOT1   ARE

NO.         REAL            IMAG.             OMEGA            ZETA

 1     -1.8445105      -.49938380        1.9109168        .96524896
 2     -1.8445105       .49938380        1.9109168        .96524896
 3     -9.3109790      0.

                    LOW ORDER NONZERO COEFFICIENT =    34.000000

DEGREE OF POLY1    IS  3          (COEFFICIENTS IN ASCENDING ORDER)
34. 38. 13. 1.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                            CP=  6.89
```

8.3 Example 3 - Multiply Two S Plane Transfer Functions

Problem: Multiply the following two s plane transfer functions

$$\frac{25}{s^2+5s+25} \quad \text{and} \quad \frac{30(\frac{s}{b}+1}{70(s)(\frac{s}{7}+1)(\frac{s}{1+j2}+1)(\frac{s}{1-j2}+1)}$$

where b=10.

The first transfer function is the same as **SPTF$_1$** from Example 1. The second transfer function is given in root form. When transfer function data is expressed in root form, gains associated with the numerator and denominator must be specified to define the transfer function. The gains used by LCAP2 correspond to the low order nonzero coefficients of the numerator and denominator polynomials if the root form were expanded out. For this example, these two gains would be 30 and 70. (A distinction is made for the low order nonzero coefficient since the low order coefficient can be zero as in the case for the denominator above.)

The FORTRAN code for this example is:

```
C      EXAMPLE 3
C
C      FIRST TRANSFER FUNCTION IS THE SAME AS SPTF1 OF EXAMPLE 1
C
C      LOAD IN ROOT DATA USING SUBROUTINE SPLDR AND POLYNOMIAL
C      ARRAYS ROOTN AND ROOTD
C
       ROOTN(1)=(1.,30.)          “real part = no. of roots
                                   imaginary part = low order nonzero coeff. of num.”
       B=10.
       ROOTN(2)=CMPLX(-B,0.)      “first root”
       ROOTD(1)=(4.,70.)          “real part = no. of roots
                                   imaginary part = low order nonzero coeff. of denom.
       ROOTD(2)=(-1.,2.)          “first root”
       ROOTD(3)=(-1.,-2.)         “second root”
       ROOTD(4)=(-7.,0.)          “third root”
       ROOTD(5)=(0.,0.)           “fourth root
       CALL SPLDR(2)              “load root data into SPTF2”
C      . . . . . . . .
C      MULTIPLY THE TWO TRANSFER FUNCTIONS
       CALL SPMPY(3,1,2)          “multiply SPTF1 and SPTF2 and store in SPTF3
C      . . . . . . .
```

The FORTRAN/PRECMP code for this example is:

```
C      EXAMPLE 3
C
C      FIRST TRANSFER FUNCTION IS THE SAME AS SPTF1 OF EXAMPLE 1
C
C      LOAD IN ROOT DATA USING SUBROUTINE SPLDR AND POLYNOMIAL
C      ARRAYS ROOTN AND ROOTD
C
       B=10.
*ROOTN 30.  B. !  GAIN, ROOT LIST
*ROOTD 70.  (-1.,2.)  -7, 0.  !  GAIN, ROOT LIST
*SPLDR 2 !  LOAD ROOT DATA INTO SPTF(2)
C      .  .  .  .  .  .  .  .  .
C      MULTIPLY THE TWO TRANSFER FUNCTIONS
*SPMPY 3 1 2 !  MULTIPLY SPTF(1) AND SPTF(2) AND STORE INTO SPTF(3)
C      .  .  .  .  .  .  .
```

The printer output for this example is:

```
                         THE ROOTS OF ROOTN   ARE

NO.        REAL            IMAG.             OMEGA            ZETA

 1     -10.000000       0.

                           THE ROOTS OF ROOTD   ARE

NO.        REAL            IMAG.             OMEGA            ZETA

 1     -1.0000000        2.0000000
 2     -1.0000000       -2.0000000
 3     -7.0000000       0.
 4     0.               0.

************************************************************
*    SPLDR - LOAD TRANSFER FUNCTION IN ROOT FORM          *
************************************************************

                      THE NUMERATOR ROOTS OF SROOT2   ARE

NO.        REAL            IMAG.             OMEGA            ZETA

 1     -10.000000       0.

                      LOW ORDER NONZERO COEFFICIENT =    30.000000
```

```
                    THE DENOMINATOR ROOTS OF SROOT2   ARE

NO.        REAL            IMAG.            OMEGA            ZETA

 1     -1.0000000        2.0000000        2.2360680        .44721360
 2     -1.0000000       -2.0000000        2.2360680        .44721360
 3     -7.0000000       0.
 4     0.               0.

                    LOW ORDER NONZERO COEFFICIENT =    70.000000

DEGREE OF NUMERATOR OF SPTF2   IS  1    (COEFFICIENTS IN ASCENDING ORDER)
30. 3.

DEGREE OF DENOMINATOR OF SPTF2   IS  4  (COEFFICIENTS IN ASCENDING ORDER)
0. 70. 38. 18. 2.
BODE GAIN =    .42857143
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                              CP=  6.90
DEGREE OF NUMERATOR OF SPTF1   IS  0    (COEFFICIENTS IN ASCENDING ORDER)
25.

DEGREE OF DENOMINATOR OF SPTF1   IS  2  (COEFFICIENTS IN ASCENDING ORDER)
25. 5. 1.

BODE GAIN =    1.0000000

                    THE NUMERATOR ROOTS OF SROOT2   ARE

NO.        REAL            IMAG.            OMEGA            ZETA

 1     -10.000000       0.

                    LOW ORDER NONZERO COEFFICIENT =    30.000000

                    THE DENOMINATOR ROOTS OF SROOT2   ARE

NO.        REAL            IMAG.            OMEGA            ZETA

 1     -1.0000000        2.0000000        2.2360680        .44721360
 2     -1.0000000       -2.0000000        2.2360680        .44721360
 3     -7.0000000       0.
 4     0.               0.

                    LOW ORDER NONZERO COEFFICIENT =    70.000000

DEGREE OF NUMERATOR OF SPTF2   IS  1    (COEFFICIENTS IN ASCENDING ORDER)
```

```
30. 3.

DEGREE OF DENOMINATOR OF SPTF2   IS  4  (COEFFICIENTS IN ASCENDING ORDER)
0. 70. 38. 18. 2.

BODE GAIN =    .42857143

***********************************************
*   SPTF3  =     SPTF1      *    SPTF2        *
***********************************************

DEGREE OF NUMERATOR OF SPTF3   IS  1    (COEFFICIENTS IN ASCENDING ORDER)
750. 75.

DEGREE OF DENOMINATOR OF SPTF3   IS  4  (COEFFICIENTS IN ASCENDING ORDER)
0. 1750. 1300. 710. 178. 28. 2.

BODE GAIN =    .42857143
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                               CP=  6.91
```

Note that if the second transfer function were expressed in a different form, the numerator and denominator nonzero coefficients would not necessarily appear explicitly. For example, if the root locus representation were used, i.e.,

$$\frac{3(s+10)}{2(s+7)(s+1+j2)(s+1-j2)}$$

then,

numerator low order nonzero coefficient = (3)(10) = 30
denominator low order nonzero coefficient = (2)(7)(1+j2)(1-j2) = 70

8.4 Example 4 - Closed Loop S Plane Transfer Function

Problem: Given open loop transfer functions G(s) and feedback transfer H(s) where

$$G(s) = \frac{15}{s^3 + 6s^2 + 5s}$$

$$H(s) = \frac{s + 1.5}{s + 15}$$

compute the closed loop transfer function F(s) where

$$F(s) = \frac{G(s)}{1 + G(s)H(s)}$$

The FORTRAN code for this example is:

```
C     EXAMPLE 4
C     LOAD IN COEFFICIENTS OF G(S)
C
      POLYN(1)=0                "degree of num."
      POLYN(2)=15.              "coefficient of s**0"
      POLYD(1)=3                "degree of denom."
      POLYD(2)=0.               "coefficient of s**0"
      POLYD(3)=5.               "coefficient of s**1"
      POLYD(4)=6.               "coefficient of s**2"
      POLYD(5)=1.               "coefficient of s**3"
      PRINT *,'G(S) IS IN SPTF4'
      CALL SPLDC(4)             "load G(s) data into SPTF4"
C     . . . . . . .
C     LOAD IN COEFFICIENTS OF H(S)
      POLYN(1)=1.               "degree of num."
      POLYN(2)=1.5              "coefficient of s**0"
      POLYN(3)=1.               "coefficient of s**1"
      POLYD(1)=1.               "degree of denom."
      POLYD(2)=15.              "coefficient of s**0"
      POLYD(3)=1.               "coefficient of s**1"
      PRINT *,'H(S) IS IN SPTF5'
      CALL SPLDC(5)             "load H(s) data into SPTF5"
C     . . . . . . .
      PRINT *,'CLOSED LOOP T.F. IS IN SPTF9'
      CALL SPCLSLP(9,4,5)       "compute closed loop transfer function"
C     . . . . . . .
```

The FORTRAN/PRECMP code for this example is:

```
C      EXAMPLE 4
C      LOAD IN COEFFICIENTS OF G(S)
C
*POLYN 0 15.  !  DEG. OF NUM., COEFF. IN ASCENDING ORDER
*POLYD 3 0.  5.  6.  1.  !  DEG. OF DENOM., COEFF. IN ASCENDING ORDER
       PRINT *,'G(S) IS IN SPTF4'
*SPLDC 4 !  LOAD G(S) DATA INTO SPTF(4)
C      . . . . . . . .
C      LOAD IN COEFFICIENTS OF H(S)
POLYN 1 1.5 1.  !  DEG. OF NUM., COEFF. IN ASCENDING ORDER
POLYD 1 15.  1.  !  DEG. OF DENOM., COEFF. IN ASCENDING ORDER
       PRINT *,'H(S) IS IN SPTF5'
*SPLDC 5 !  LOAD H(S) DATA INTO SPTF(5)
C      . . . . . . . .
       PRINT *,'CLOSED LOOP T.F. IS IN SPTF9'
*SPCLSLP 9 4 5 !  COMPUTE CLOSED LOOP TRANSFER FUNCTION
C      . . . . . . . .
```

The printer output for this example is:

```
G(S) IS IN SPTF4

DEGREE OF POLYN   IS  0          (COEFFICIENTS IN ASCENDING ORDER)
15.

DEGREE OF POLYD   IS  3          (COEFFICIENTS IN ASCENDING ORDER)
0. 5. 6. 1.

************************************************************
*    SPLDC - LOAD TRANSFER FUNCTION IN COEFFICIENT FORM    *
************************************************************

DEGREE OF NUMERATOR OF SPTF4   IS  0   (COEFFICIENTS IN ASCENDING ORDER)
15.

DEGREE OF DENOMINATOR OF SPTF4   IS  3  (COEFFICIENTS IN ASCENDING ORDER)
0. 5. 6. 1.

BODE GAIN =    3.0000000
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                            CP=   6.92
H(S) IS IN SPTF5

DEGREE OF POLYN   IS  1          (COEFFICIENTS IN ASCENDING ORDER)
```

```
1.5 1.

DEGREE OF POLYD   IS  1          (COEFFICIENTS IN ASCENDING ORDER)
15. 1.

************************************************************
*    SPLDC - LOAD TRANSFER FUNCTION IN COEFFICIENT FORM    *
************************************************************

DEGREE OF NUMERATOR OF SPTF5   IS  1    (COEFFICIENTS IN ASCENDING ORDER)
1.5 1.

DEGREE OF DENOMINATOR OF SPTF5   IS  1  (COEFFICIENTS IN ASCENDING ORDER)
15. 1.

BODE GAIN =    .10000000E+00
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                               CP=   6.92
CLOSED LOOP T.F. IS IN SPTF9

DEGREE OF NUMERATOR OF SPTF4   IS  0    (COEFFICIENTS IN ASCENDING ORDER)
15.

DEGREE OF DENOMINATOR OF SPTF4   IS  3  (COEFFICIENTS IN ASCENDING ORDER)
0. 5. 6. 1.

BODE GAIN =    3.0000000

DEGREE OF NUMERATOR OF SPTF5   IS  1    (COEFFICIENTS IN ASCENDING ORDER)
1.5 1.

DEGREE OF DENOMINATOR OF SPTF5   IS  1  (COEFFICIENTS IN ASCENDING ORDER)
15. 1.

BODE GAIN =    .10000000E+00

***********************************************
*   SPTF9   =      SPTF4      /( 1. + SPTF5   ) *
***********************************************

DEGREE OF NUMERATOR OF SPTF9   IS  1    (COEFFICIENTS IN ASCENDING ORDER)
225. 15.

DEGREE OF DENOMINATOR OF SPTF9   IS  4  (COEFFICIENTS IN ASCENDING ORDER)
22.5 90. 95. 21. 1.

BODE GAIN =    10.000000
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

8.5 Example 5 - S Plane Root Locus

Problem: Compute the root locus for the following transfer function

$$\frac{1.8(\frac{s}{1})}{s(\frac{s}{2}+1)(\frac{s}{3}+1)(\frac{s}{5-j1})(\frac{s}{5+j1}+1)}$$

by varying the nominal gain from .125 to 2.

The FORTRAN code for this example is:

```
C      EXAMPLE 5
C      LOAD IN ROOT DATA FOR TRANSFER FUNCTION
C
       ROOTN(1)=(1.,1.8)         "real part = no. of roots
                                  imaginary part = low order nonzero coeff. of num."
       ROOTN(2)=(-1.,0.)         "first root"
       ROOTD(1)=(5.,1.)          "real part = no. of roots
                                  imaginary part = low order nonzero coeff. of denom."
       ROOTD(2)=(-5.,-1.)        "first root"
       ROOTD(3)=(-5.,1.)         "second root"
       ROOTD(4)=(-3.,0.)         "third root"
       ROOTD(5)=(-2.,0.)         "fourth root"
       ROOTD(6)=(0.,0.)          "fifth root"
       CALL SPLDR(10)            "data stored in SPTF10
C      . . . . . . .
C      ENTER ROOT LOCUS PARAMETERS FOR USE WITH SLOCI
C
       NLOCI=2                   "number of values of KGAIN to be entered"
       KGAIN(1)=.125             "KGAIN(1) = first gain value to be used,
       KGAIN(2)=2.               "KGAIN(NLOCI)=last gain value to be used"
       KFLG=0                     .EQ.0 to increment gain by multiplying by KDELT
                                  .NE.0 to increment gain by adding by KDELT"
       KDELT=2                   "value for changing gains (preset=1.E-4 so that no
                                  additional gains are computed by the program"
       GRAFP=1                   ".NE.0 (preset=1) for low resolution printer plot"
       HRDCPY=1                  ".NE.0 (preset=0) for high resolution hardcopy plot"
       RLXMIN=-9                 "min. x axis for plot"
       RLXMAX=1                  "max. x axis for plot"
                                 "auto. scaling for x axis if RLXMN=RLXMX"
       RLYMIN=-1                 "min. y axis for plot"
       RLYMAX=9                  "max. y axis for plot"
                                 "auto. scaling for y axis if RLYMN=RLYMX"
       TITLE1 = 'EXAMPLE 5 S PLANE ROOT LOCUS'
       CALL SLOCI(10)            "compute root loci of SPTF5"
C      . . . . . . .
```

The FORTRAN/PRECMP code for this example is:

```
C      EXAMPLE 5
C      LOAD IN ROOT DATA FOR TRANSFER FUNCTION
C
*ROOTN 1.8 -1 !  GAIN, ROOT LIST
*ROOTD 1.  (-5.,1.)  -3.  -2.  0.  !  GAIN, ROOT LIST
*SPLDR 10 !  LOAD ROOT DATA INTO SPTF(10)
C      .  .  .  .  .  .  .
C      ENTER ROOT LOCUS PARAMETERS FOR USE WITH SLOCI
C
       NLOCI=2                    "number of values of KGAIN to be entered"
KGAIN .125 2.  !  KGAIN(1)=FIRST GAIN,KGAIN(NLOCI)=LAST GAIN
       KFLG=0                     .EQ.0 to increment gain by multiplying by KDELT
                                  .NE.0 to increment gain by adding by KDELT"
       KDELT=2                    "value for changing gains (preset=1.E-4 so that no
                                   additional gains are computed by the program"
       GRAFP=1                    ".NE.0 (preset=1) for low resolution printer plot"
       HRDCPY=1                   ".NE.0 (preset=0) for high resolution hardcopy plot"
       RLXMIN=-9                  "min. x axis for plot"
       RLXMAX=1                   "max. x axis for plot"
                                  "auto. scaling for x axis if RLXMN=RLXMX"
       RLYMIN=-1                  "min. y axis for plot"
       RLYMAX=9                   "max. y axis for plot"
                                  "auto. scaling for y axis if RLYMN=RLYMX"
       TITLE1 = 'EXAMPLE 5 S PLANE ROOT LOCUS'
*SLOCI 10 !  COMPUTE ROOT LOCI OF SPTF(5)
C      .  .  .  .  .  .  .
```

The printer output for this example is:

```
                           THE ROOTS OF ROOTN   ARE

NO.          REAL              IMAG.              OMEGA             ZETA

 1      -1.0000000       0.

                           THE ROOTS OF ROOTD   ARE

NO.          REAL              IMAG.              OMEGA             ZETA

 1      -5.0000000       -1.0000000
 2      -5.0000000        1.0000000
 3      -3.0000000       0.
 4      -2.0000000       0.
 5      0.               0.

************************************************************
*    SPLDR - LOAD TRANSFER FUNCTION IN ROOT FORM           *
************************************************************

                    THE NUMERATOR ROOTS OF SROOT10  ARE

NO.          REAL              IMAG.              OMEGA             ZETA

 1      -1.0000000       0.

                    LOW ORDER NONZERO COEFFICIENT =    1.8000000

                    THE DENOMINATOR ROOTS OF SROOT10  ARE

NO.          REAL              IMAG.              OMEGA             ZETA

 1      -5.0000000       -1.0000000         5.0990195         .98058068
 2      -5.0000000        1.0000000         5.0990195         .98058068
 3      -3.0000000       0.
 4      -2.0000000       0.
 5      0.               0.

                    LOW ORDER NONZERO COEFFICIENT =    1.0000000

DEGREE OF NUMERATOR OF SPTF10 IS  1    (COEFFICIENTS IN ASCENDING ORDER)
1.8 1.8

DEGREE OF DENOMINATOR OF SPTF10 IS  5  (COEFFICIENTS IN ASCENDING ORDER)
0. 1. 1.217948717949 .525641025641 .09615384615385 .00641025641025G
```

```
BODE GAIN =    1.8000000
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                                CP=   6.95
                    THE NUMERATOR ROOTS OF SROOT10  ARE

NO.        REAL            IMAG.            OMEGA            ZETA

 1     -1.0000000       0.

                    LOW ORDER NONZERO COEFFICIENT =    1.8000000

                    THE DENOMINATOR ROOTS OF SROOT10  ARE

NO.        REAL            IMAG.            OMEGA            ZETA

 1     -5.0000000      -1.0000000       5.0990195        .98058068
 2     -5.0000000       1.0000000       5.0990195        .98058068
 3     -3.0000000       0.
 4     -2.0000000       0.
 5      0.              0.

                    LOW ORDER NONZERO COEFFICIENT =    1.0000000

DEGREE OF NUMERATOR OF SPTF10  IS  1   (COEFFICIENTS IN ASCENDING ORDER)
1.8 1.8

DEGREE OF DENOMINATOR OF SPTF10  IS  5  (COEFFICIENTS IN ASCENDING ORDER)
0. 1. 1.217948717949 .525641025641 .09615384615385 .006410256410256

BODE GAIN =    1.8000000

************************************************************
*     SLOCI - ROOT LOCUS OF S-PLANE TRANSFER               *
*             FUNCTION 1                                   *
************************************************************

                           THE OPEN LOOP ZEROES ARE

NO.        REAL            IMAG.            OMEGA            ZETA

 1     -1.0000000       0.

                           THE OPEN LOOP POLES ARE

NO.        REAL            IMAG.            OMEGA            ZETA

 1     -5.0000000      -1.0000000       5.0990195        .98058068
```

```
  2     -5.0000000        1.0000000        5.0990195        .98058068
  3     -3.0000000       0.
  4     -2.0000000       0.
  5     0.               0.
----------------------------------------------------------------------

CLOSED LOOP POLES FOR GAIN =    .12500000      (GAIN NO.  1) ARE

NO.         REAL             IMAG.             OMEGA             ZETA

 1      -1.7947341       -1.1077310        2.1090611        .85096357
 2      -1.7947341        1.1077310        2.1090611        .85096357
 3      -5.5892217       -1.6614254        5.8309290        .95854737
 4      -5.5892217        1.6614254        5.8309290        .95854737
 5      -.23208832       0.
----------------------------------------------------------------------

CLOSED LOOP POLES FOR GAIN =    .25000000      (GAIN NO.  2) ARE

NO.         REAL             IMAG.             OMEGA             ZETA

 1      -1.4171985       -1.4418858        2.0217532        .70097500
 2      -1.4171985        1.4418858        2.0217532        .70097500
 3      -5.8577544       -1.9606112        6.1771582        .94829277
 4      -5.8577544        1.9606112        6.1771582        .94829277
 5      -.45009415       0.
----------------------------------------------------------------------

CLOSED LOOP POLES FOR GAIN =    .50000000      (GAIN NO.  3) ARE

NO.         REAL             IMAG.             OMEGA             ZETA

 1      -.94735061       -1.9000842        2.1231564        .44619915
 2      -.94735061        1.9000842        2.1231564        .44619915
 3      -6.1975818        2.3343777        6.6226384        .93581763
 4      -6.1975818       -2.3343777        6.6226384        .93581763
 5      -.71013518       0.
----------------------------------------------------------------------

CLOSED LOOP POLES FOR GAIN =    1.0000000      (GAIN NO.  4) ARE

NO.         REAL             IMAG.             OMEGA             ZETA

 1      -.44922549       -2.4680974        2.5086468        .17907084
 2      -.44922549        2.4680974        2.5086468        .17907084
 3      -6.6183282        2.7904455        7.1825382        .92144699
 4      -6.6183282       -2.7904455        7.1825382        .92144699
 5      -.86489263       0.
```

```
-----------------------------------------------------------------------

CLOSED LOOP POLES FOR GAIN =    2.0000000      (GAIN NO.  5) ARE

NO.        REAL              IMAG.             OMEGA            ZETA

 1       .10025104        -3.1088677        3.1104837       -.32230049E-01
 2       .10025104         3.1088677        3.1104837       -.32230049E-01
 3     -7.1323100         -3.3395771        7.8754442        .90563907
 4     -7.1323100          3.3395771        7.8754442        .90563907
 5     -.93588209         0.
     -----------------------------------------------------------------
```

```
ROOT LOCUS                                                    09/07/88
EXAMPLE 5 S PLANE ROOT LOCUS

  10.2                                                            I
  9.90                                                            I
  9.60                                                            I
  9.30                                                            I
  9.00                                                            I
  8.70                                                            I
  8.40                                                            I
  8.10                                                            I
  7.80                                                            I
  7.50                                                            I
  7.20                                                            I
  6.90                                                            I
  6.60                                                            I
  6.30                                                            I
  6.00                                                            I
  5.70                                                            I
  5.40                                                            I
  5.10                                                            I
  4.80                                                            I
  4.50                                                            I
  4.20                                                            I
  3.90                                                            I
  3.60                                                            I
  3.30                 *                                          I
  3.00                                                            I*
  2.70                    *                                       I
  2.40                       *                                 *  I
  2.10                         *                                  I
  1.80                          *                           *     I
  1.50                                                   *        I
  1.20                                                 *          I
  .900                                                            I
  .600                                                            I
  .300                                                            I
  0.        ------------------------------------------------****-*I------
  -.300                                                           I
  -.600                                                           I
  -.900                                                           I
  -1.20                                                           I
  -1.50                                                           I
  -1.80                                                           I
  -2.10                                                           I
  -2.40                                                           I
            I     I     I     I     I     I     I     I     I     I     I
         -9.00 -8.00 -7.00 -6.00 -5.00 -4.00 -3.00 -2.00 -1.00   .00  1.00
```

8.6 Example 6 - Inverse Laplace Transform and Time Response

Problem: Find the inverse Laplace transform and the step response of the following transfer function

$$\frac{432s + 4320}{s^4 + 35s^3 + 345s^2 + 1008s + 2160}$$

Plot the response in increments of .05 seconds from 0 to 5 seconds.

The FORTRAN code for this example is:

```
C      EXAMPLE 6
C      LOAD COEFFICIENT DATA FOR TRANSFER FUNCTION
C
       POLYN(1)=1.                 "degree of numerator"
       POLYN(2)=4320.              "coefficient of s**0"
       POLYN(3)=432.               "coefficient of s**1"
       POLYD(1)=4                  "degree of denominator"
       POLYD(2)=2160.              "coefficient of s**0"
       POLYD(3)=1008.              "coefficient of s**1"
       POLYD(4)=345.               "coefficient of s**2"
       POLYD(5)=35.                "coefficient of s**3"
       POLYD(6)=1.                 "coefficient of s**4"
       CALL SPLDC(11)              "load coefficient data into SPTF11
C      . . . . . . . .
C      ENTER PARAMETERS FOR TIME RESPONSE
C
       TTYPE=1.                    ".NE.0 (preset=1) for step response
                                    .EQ.0 for impulse response "
       TMAGN=1.                    "magnitude of the input"
       TZERO=0.                    "starting time for evaluating the response"
       TEND=5.                     "end time for evaluating the response"
       TDELT=.05                   "time increment for evaluating the response"
       GRAFP=1                     ".NE.0 (preset=1) for low resolution printer plot"
       HRDCPY=1                    ".NE.0 (preset=0) for high resolution hardcopy plot"
       TITLE1 =
      +'EXAMPLE 6 INVERSE LAPLACE TRANSFORM AND TIME RESPONSE'
       CALL STIME(11)              "compute inverse Laplace transform and time response"
C      . . . . . . . .
```

The FORTRAN/PRECMP code for this example is:

```
C      EXAMPLE 6
C      LOAD COEFFICIENT DATA FOR TRANSFER FUNCTION
C
*POLYN 1 4320 432 !  DEG. OF NUM., COEFF. IN ASCENDING ORDER
*POLYD 4 2160.  1000.  345.  35.  1.  !  DEG. OF DENOM., COEFF. IN ASCENDING ORD.
*SPLDC 11 !  LOAD COEFFICIENT DATA INTO SPTF(11)
C      . . . . . . .
C      ENTER PARAMETERS FOR TIME RESPONSE
C
       TTYPE=1.                          ".NE.0 (preset=1) for step response
                                          .EQ.0 for impulse response "
       TMAGN=1.                          "magnitude of the input"
       TZERO=0.                          "starting time for evaluating the response"
       TEND=5.                           "end time for evaluating the response"
       TDELT=.05                         "time increment for evaluating the response"
       GRAFP=1                           ".NE.0 (preset=1) for low resolution printer plot"
       HRDCPY=1                          ".NE.0 (preset=0) for high resolution hardcopy plot"
       TITLE1 =
      +'EXAMPLE 6 INVERSE LAPLACE TRANSFORM AND TIME RESPONSE'
*STIME 11 !  COMPUTE INVERSE LAPLACE TRANSFORM AND TIME RESPONSE
C      . . . . . . .
```

The printer output for this example is:

```
DEGREE OF POLYN   IS  1          (COEFFICIENTS IN ASCENDING ORDER)
4320. 432.

DEGREE OF POLYD   IS  4          (COEFFICIENTS IN ASCENDING ORDER)
2160. 1008. 345. 35. 1.

************************************************************
*    SPLDC - LOAD TRANSFER FUNCTION IN COEFFICENT FORM     *
************************************************************

DEGREE OF NUMERATOR OF SPTF11  IS  1    (COEFFICIENTS IN ASCENDING ORDER)
4320. 432.

DEGREE OF DENOMINATOR OF SPTF11  IS  4  (COEFFICIENTS IN ASCENDING ORDER)
2160. 1008. 345. 35. 1.

BODE GAIN =    2.0000000
```

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                                 CP=  7.05
                    THE NUMERATOR ROOTS OF SROOT11  ARE

NO.        REAL            IMAG.            OMEGA            ZETA

 1     -10.000000       0.

                    LOW ORDER NONZERO COEFFICIENT =    4320.0000

                    THE DENOMINATOR ROOTS OF SROOT11  ARE

NO.        REAL            IMAG.            OMEGA            ZETA

 1     -1.5000000      -2.5980762       3.0000000        .50000000
 2     -1.5000000       2.5980762       3.0000000        .50000000
 3     -12.000000       0.
 4     -20.000000       0.

                    LOW ORDER NONZERO COEFFICIENT =    2160.0000

DEGREE OF NUMERATOR OF SPTF11  IS  1    (COEFFICIENTS IN ASCENDING ORDER)
4320. 432.

DEGREE OF DENOHINATOR OF SPTF11  IS  4  (COEFFICIENTS IN ASCENDING ORDER)
2160. 1008. 345. 35. 1.

BODE GAIN =    2.0000000

**************************************************************
*     STIME = TIME RESPONSE OF S-PLANE TRANSFER              *
*             FUNCTION 11                                    *
**************************************************************

COMPUTE STEP RESPONSE IF TTYPE.NE.0., (TTYPE =  1.0 )
OTHERWISE COMPUTE IMPULSE RESPONSE.

SCALE OUTPUT BY TMAGN, (TMAGN =   1.00000     )

 NO.                ROOT                 PARTIAL FRACTION COEFFICIENT

 1     -1.5000000      -2.5980762       -.99977959       -.69735056
 2     -1.5000000       2.5980762       -.99977959        .69735056
 3     -12.000000       0.               .76923077E-01   0.
 4     -20.000000       0.              -.77363897E-01   0
 5     0.               0.               2.0000000       0.

ANALYTICAL SOLUTION IS THE SUMMATION OF THE FOLLOWING  4 TERMS
```

```
                                                     (   2.0000     ) * T** 0
((-2.000     )*(COS( 2.60     *T))+(-1.395     )*(SIN( 2.60    *T)))*E**(-1.50*T)
                                              (  .76923E-01) * E**( -12.000 * T)
                                              ( -.77364E-01) * E**( -20.000 * T)

                                *** TIME RESPONSE ***

  TIME          VALUE          TIME          VALUE          TIME          VALUE

 0.            0.             .50000E-01    .66945E-02    .10000        .41037E-01
  .15000        .10879        .20000        .20673        .25000        .32913
  .30000        .46993        .35000        .62345        .40000        .78455
  .45000        .94871        .50000       1.1120         .55000       1.2711
  .60000       1.4233         .65000       1.5664         .70000       1.6985
  .75000       1.8186         .80000       1.9257         .85000       2.0194
  .90000       2.0997         .95000       2.1667        1.0000        2.2209
 1.0500        2.2630        1.1000        2.2937        1.1500        2.3140
 1.2000        2.3249        1.2500        2.3276        1.3000        2.3230
 1.3500        2.3123        1.4000        2.2966        1.4500        2.2769
 1.5000        2.2542        1.5500        2.2293        1.6000        2.2032
 1.6500        2.1764        1.7000        2.1497        1.7500        2.1235
 1.8000        2.0985        1.8500        2.0749        1.9000        2.0530
 1.9500        2.0330        2.0000        2.0152        2.0500        1.9995
 2.1000        1.9860        2.1500        1.9747        2.2000        1.9655
 2.2500        1.9582        2.3000        1.9529        2.3500        1.9493
 2.4000        1.9472        2.4500        1.9466        2.5000        1.9471
 2.5500        1.9487        2.6000        1.9511        2.6500        1.9542
 2.7000        1.9578        2.7500        1.9618        2.8000        1.9661
 2.8500        1.9704        2.9000        1.9748        2.9500        1.9791
 3.0000        1.9832        3.0500        1.9871        3.1000        1.9907
 3.1500        1.9940        3.2000        1.9970        3.2500        1.9996
 3.3000        2.0019        3.3500        2.0038        3.4000        2.0054
 3.4500        2.0066        3.5000        2.0075        3.5500        2.0082
 3.6000        2.0086        3.6500        2.0087        3.7000        2.0086
 3.7500        2.0084        3.8000        2.0081        3.8500        2.0076
 3.9000        2.0070        3.9500        2.0063        4.0000        2.0057
 4.0500        2.0050        4.1000        2.0042        4.1500        2.0035
 4.2000        2.0029        4.2500        2.0022        4.3000        2.0016
 4.3500        2.0011        4.4000        2.0006        4.4500        2.0001
 4.5000        1.9998        4.5500        1.9994        4.6000        1.9992
 4.6500        1.9990        4.7000        1.9988        4.7500        1.9987
 4.8000        1.9986        4.8500        1.9986        4.9000        1.9986
 4.9500        1.9986        5.0000        1.9987
```

```
TIME RESPONSE                                                  09/07/88
EXAMPLE 6 INVERSE LAPLACE TRANSFORM AND TIME RESPONSE

 3.30     I
 3.20     I
 3.10     I
 3.00     I
 2.90     I
 2.80     I
 2.70     I
 2.60     I
 2.50     I
 2.40     I
 2.30     I            ******
 2.20     I          **      **
 2.10     I          *        ****
 2.00     I         *            ******  ******************************
 1.90     I         *                 ****
 1.80     I        *
 1.70     I       *
 1.60     I       *
 1.50     I
 1.40     I      *
 1.30     I      *
 1.20     I
 1.10     I     *
 1.00     I
 .900     I    *
 .800     I    *
 .700     I
 .600     I   *
 .500     I   *
 .400     I
 .300     I  *
 .200     I *
 .100     I *
0.        **-----------------------------------------------------------
-.100E+00 I
-.200     I
-.300     I
-.400     I
-.500     I
-.600     I
-.700     I
-.800     I
-.900     I
          I     I     I     I     I     I     I     I     I     I     I
         .00   .50  1.00  1.50  2.00  2.50  3.00  3.50  4.00  4.50  5.00
```

8.7 Example 7 - Inverse Z Transform

Problem: Find the inverse z transform and the step response of the following transfer function

$$\frac{a_4 z^4 + a_3 z^3 + a_2 z^2 + a_1 z + a_0}{b_4 z^4 + b_3 z^3 + b_2 z^2 + b_1 z^1 + b_0}$$

$$\begin{array}{rll}
\text{where, } a_0 = 0 & & b_0 = 0.173773943 \\
a_1 = \text{-}3.64646532\text{E-}3 & & b_1 = \text{-}1.16046330 \\
a_2 = 4.90388971\text{E-}4 & & b_2 = 2.74918160 \\
a_3 = 9.10367045\text{E-}3 & & b_3 = \text{-}2.75654438 \\
a_4 = 0 & & b_4 = 1.
\end{array}$$

and the sampling period is .05 seconds. Plot the results from t=0 to t=2.5 seconds.

Since the inverse is computed by recursive evaluation of a difference equation the results will be inaccurate for higher order transfer functions. Results for typical transfer functions whose order is less than 10~15 are very good.

The FORTRAN code for this example is:

```
C      EXAMPLE 7
C      LOAD IN COEFFICIENT DATA INTO ZPTF1
C
       POLYN(1)=3.                  “degree of numerator”
       POLYN(2)=0.                  “coefficient of z**0”
       POLYN(3)=-3.64646532E-3      “coefficient of z**1”
       POLYN(4)=4.90388971E-4       “coefficient of z**2”
       POLYN(5)=9.10367045E-3       “coefficient of z**3”
       POLYD(1)=4.                  “degree of denominator”
       POLYD(2)=.173773943          “coefficient of z**0”
       POLYD(3)=-1.16046330         “coefficient of z**1”
       POLYD(4)=2.74918160          “coefficient of z**2”
       POLYD(5)=-2.75654438         “coefficient of z**3”
       POLYD(6)=1.                  “coefficient of z**4”
       CALL ZPLDC(1)
C      . . . . . . . .
C      ENTER TIME RESPONSE PARAMETERS FOR USE WITH ZTIME
C
       TTYPE=1                      “.NE.0 (preset=1) for step response
                                     .EQ.0 for impulse response ”
       TMAGN=1.                     “magnitude of the input”
       TEND=2.5                     “end time for evaluating the response”
       SAMPT=.05                    “sampling period”
       TITLE1 =
      +'EXAMPLE 7 INVERSE Z TRANSFORM AND TIME RESPONSE'
       GRAFP=1                      “.NE.0 (preset=1) for low resolution printer plot”
       HRDCPY=1                     “.NE.0 (preset=0) for high resolution printer plot”
      CALL ZTIME(1)                "compute inverse z transform of ZPTF1
C     . . . . . . . .
```

The FORTRAN/PRECMP code for this example is:

```
C     EXAMPLE 7
C     LOAD IN COEFFICIENT DATA INTO ZPTF1
C
*POLYN 3 0.  -3.64646532e-3 4.90388971e-4 9.10367045E-3
*POLYD 4 .173773943 -1.16046330 2.74918160 -2.75654438 1.
*ZPLDC 1 !  LOAD COEFFICIENT DATA INTO ZPTF(1)
C     . . . . . . . .
C     ENTER TIME RESPONSE PARAMETERS FOR USE WITH ZTIME
C
      TTYPE=1                      ".NE.0 (preset=1) for step response
                                    .EQ.0 for impulse response "
      TMAGN=1.                     "magnitude of the input"
      TEND=2.5                     "end time for evaluating the response"
      SAMPT=.05                    "sampling period"
      TITLE1 =
     +'EXAMPLE 7 INVERSE Z TRANSFORM AND TIME RESPONSE'
      GRAFP=1                      ".NE.0 (preset=1) for low resolution printer plot"
      HRDCPY=1                     ".NE.0 (preset=0) for high resolution printer plot"
*ZTIME 1 !  INVERSE Z TRANSFORM OF ZPTF(1)
C     . . . . . . . .
```

The printer output for this example is:

```
DEGREE OF POLYN    IS  3          (COEFFICIENTS IN ASCENDING ORDER)
0. -.00364646532 .000490388971 .00910367045

DEGREE OF POLYD    IS  4          (COEFFICIENTS IN ASCENDING ORDER)
.173773943 -1.1604633 2.7491816 -2.75654438 1.

************************************************************
*   ZPLDC - LOAD TRANSFER FUNCTION IN COEFFICENT FORM      *
************************************************************

DEGREE OF NUMERATOR OF ZPTF1   IS  3   (COEFFICIENTS IN ASCENDING ORDER)
0. -.00364646532 .000490388971 .00910367045
```

```
DEGREE OF DENOMINATOR OF ZPTF1   IS  4  (COEFFICIENTS IN ASCENDING ORDER)
.173773943 -1.1604633 2.7491816 -2.75654438 1.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                               CP=  7.14
DEGREE OF NUMERATOR OF ZPTF1   IS  3    (COEFFICIENTS IN ASCENDING ORDER)
0. -.00364646532 .000490388971 .00910367045

DEGREE OF DENOMINATOR OF ZPTF1   IS  4  (COEFFICIENTS IN ASCENDING ORDER)
.173773943 -1.1604633 2.7491816 -2.75654438 1.

************************************************************
*     ZTIME = TIME RESPONSE OF Z-PLANE TRANSFER            *
*             FUNCTION 1                                   *
************************************************************

COMPUTE STEP RESPONSE IF TTYPE.NE.0., (TTYPE =  1.0 )
OTHERWISE COMPUTE IMPULSE RESPONSE.

SCALE OUTPUT BY TMAGN, (TMAGN =   1.00000     )

FINAL VALUE =    .999955

                         *** TIME RESPONSE ***

  TIME          VALUE        TIME          VALUE        TIME          VALUE

0.            0.           .50000E-01   .91037E-02   .10000        .34689E-01
 .15000        .76541E-01  .20000       .13214       .25000        .19843
 .30000        .27247      .35000       .35152       .40000        .43319
 .45000        .51536      .50000       .59622       .55000        .67426
 .60000        .74822      .65000       .81715       .70000        .88028
 .75000        .93712      .80000       .98736       .85000        1.0309
 .90000        1.0677      .95000       1.0979       1.0000        1.1219
 1.0500        1.1400      1.1000       1.1527       1.1500        1.1604
 1.2000        1.1637      1.2500       1.1631       1.3000        1.1592
 1.3500        1.1526      1.4000       1.1436       1.4500        1.1330
 1.5000        1.1210      1.5500       1.1082       1.6000        1.0949
 1.6500        1.0814      1.7000       1.0682       1.7500        1.0554
 1.8000        1.0432      1.8500       1.0318       1.9000        1.0213
 1.9500        1.0118      2.0000       1.0034       2.0500        .99614
 2.1000        .98994      2.1500       .98481       2.2000        .98072
 2.2500        .97759      2.3000       .97537       2.3500        .97397
 2.4000        .97330      2.4500       .97329       2.5000        .97383
```

```
TIME RESPONSE                                                     09/07/88
EXAMPLE 7 INVERSE Z TRANSFORM AND TIME RESPONSE

  1.20     I
  1.17     I                           ****
  1.14     I                       **      * **
  1.11     I                      **           **
  1.08     I                     *               * **
  1.05     I                                         **
  1.02     I                   *                       * **
  .990     I                  *                            *** ****
  .960     I                                                       * ***
  .930     I                 *
  .900     I
  .870     I                *
  .840     I
  .810     I               *
  .780     I
  .750     I             *
  .720     I
  .690     I
  .660     I            *
  .630     I
  .600     I           *
  .570     I
  .540     I
  .510     I          *
  .480     I
  .450     I
  .420     I         *
  .390     I
  .360     I       *
  .330     I
  .300     I
  .270     I      *
  .240     I
  .210     I     *
  .180     I
  .150     I
  .120     I    *
  .900E-01 I   *
  .600E-01 I
  .300E-01 I *
 0.        **-----------------------------------------------------------
 -.300E-01 I
 -.600E-01 I
           I     I     I     I     I     I     I     I     I     I     I
          .00   .25   .50   .75  1.00  1.25  1.50  1.75  2.00  2.25  2.50
```

8.8 Example 8 - Z Transform of an S Plane Transfer Function

Problem: Compute the z transform of the following transfer function with a zero-order and a delay of .008 seconds. The sampling period is .08 seconds.

$$\frac{2900s + 2900}{s^5 + 4s^4 + 124s^3 + 363s^2}$$

The partial fraction method is used to compute the z transform. The algorithm used to compute the partial fraction expansion requires that (1) there are no multiple poles other than those at the origin and (2) the degree of the numerator must not be greater than the number of poles which are not at the origin. Up to 5 poles at the origin are allowed (this includes the ones from the zero-order hold if selected).

The FORTRAN code for this example is:

```
C     EXAMPLE 8
C     LOAD COEFFICIENT DATA INTO SPTF12
C
      POLYN(1)=1                  “degree of numerator”
      POLYN(2)=2900.              “coefficient of s**0”
      POLYN(3)=2900.              “coefficient of s**1”
      POLYD(1)=5.                 “degree of denominator.”
      POLYD(2)=0.                 “coefficient of s**0”
      POLYD(3)=0.                 “coefficient of s**1”
      POLYD(4)=363.               “coefficient of s**2”
      POLYD(5)=124.               “coefficient of s**3”
      POLYD(6)=4.                 “coefficient of s**4”
      POLYD(7)=1.                 “coefficient of s**5”
      CALL SPLDC(12)              “load coefficient data into SPTF12”
C . . . . . . . .
C ENTER PARAMETERS FOR USE WITH SZXFM
C
      SAMPT=.08                   “sampling period”
      DELAY=.008                  “time delay (preset=0.)”
                                   (enter a negative value for time advance)
      ZOH=1                       “.NE.0 (preset=1) for inclusion of zero order hold”
      CALL SZXFM(2,12)            “compute z transform of SPTF12 and store into ZPTF2”
C     . . . . . . . .
```

The FORTRAN/PRECMP code for this example is:

```
C      EXAMPLE 8
C      LOAD COEFFICIENT DATA INTO SPTF12
C
*POLYN 1 2900.  2900.  !  NUMERATOR DATA
*POLYD 5 0.  0.  363.  124.  4.  1.  !  DENOMINATOR DATA
*SPLDC 12 !  LOAD COEFFICIENT DATA INTO SPTF(12)
C . . . . . . . .
C ENTER PARAMETERS FOR USE WITH SZXFM
C
      SAMPT=.08                  "sampling period"
      DELAY=.008                 "time delay (preset=0.)"
                                  (enter a negative value for time advance)
      ZOH=1                      ".NE.0 (preset=1) for inclusion of zero order hold"
*SZXFM 2 12 !  COMPUTE Z TRANSFORM OF SPTF(12), STORE IN ZPTF(2)
C      . . . . . . . .
```

The printer output for this example is:

```
DEGREE OF POLYN   IS  1          (COEFFICIENTS IN ASCENDING ORDER)
2900. 2900.

DEGREE OF POLYD   IS  5          (COEFFICIENTS IN ASCENDING ORDER)
0. 0. 363. 124. 4. 1.

************************************************************
*    SPLDC - LOAD TRANSFER FUNCTION IN COEFFICENT FORM     *
************************************************************

DEGREE OF NUMERATOR OF SPTF12  IS  1    (COEFFICIENTS IN ASCENDING ORDER)
2900. 2900.

DEGREE OF DENOMINATOR OF SPTF12  IS  5  (COEFFICIENTS IN ASCENDING ORDER)
0. 0. 363. 124. 4. 1.

BODE GAIN =    7.9889807
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                          CP=  7.19
                    THE NUMERATOR ROOTS OF SROOT12  ARE


NO.        REAL             IMAG.              OMEGA            ZETA
```

```
 1     -1.0000000       0.

                    LOW ORDER NONZERO COEFFICIENT =    2900.0000

                    THE DENOMINATOR ROOTS OF SROOT12  ARE

NO.        REAL            IMAG.             OMEGA            ZETA

 1     -.50000000       -10.988630        11.000000        .45454545E-01
 2     -.50000000        10.988630        11.000000        .45454545E-01
 3     -3.0000000       0.
 4     0.               0.
 5     0.               0.

                    LOW ORDER NONZERO COEFFICIENT =    363.00000

DEGREE OF NUMERATOR OF SPTF12  IS  1    (COEFFICIENTS IN ASCENDING ORDER)
2900. 2900.

DEGREE OF DENOMINATOR OF SPTF12  IS  5  (COEFFICIENTS IN ASCENDING ORDER)
0. 0. 363. 124. 4. 1.

BODE GAIN =    7.9889807

***********************************************
*  ZROOT2  =    SZXFM OF        ZROOT12       *
***********************************************

***********************************************
*   ZPTF2  =    SZXFM OF         ZPTF12       *
***********************************************

SAMPLING PERIOD, SAMPT =   .0800     DELAY =   .0080     ZOH = 1

 NO.              ROOT          PARTIAL FRACTION EXPANSION COEFFICIENT

 1     -.50000000       -10.988630        .96684606E-01   -.40456452E-02
 2     -.50000000        10.988630        .96684606E-01    .40456452E-02
 3     -3.0000000       0.                1.6914552       0.
 4     0.               0.               -1.8848244       0.
 5     0.               0.                5.2599625       0.
 6     0.               0.                7.9889807       0.

                    THE NUMERATOR ROOTS OF ZROOT2   ARE

NO.        REAL            IMAG.             OMEGA            ZETA

 1     -12.268933       0.
 2     -.67135127E-04   0.
 3      .92311634       0.
 4     -.13469859       0.
 5     -1.1567519       0.

                         LOW ORDER NONZERO COEFFICIENT =    -.23106188E-04

                         THE DENOMINATOR ROOTS OF ZROOT2   ARE

NO.        REAL            IMAG.            OMEGA             ZETA

 1      .61284138       -.73996066       .96078944        -.63785191
 2      .61284138        .73996066       .96078944        -.63785191
 3      .78662786       0.
 4      1.0000000       0.
 5      1.0000000       0.
 6     0.               0.

                         LOW ORDER NONZERO COEFFICIENT =    -46.473538

DEGREE OF NUMERATOR OF ZPTF2   IS  5    (COEFFICIENTS IN ASCENDING ORDER)
-.00002310618764969 -.3443427037531 -2.507846683739 .679417062974 2.464719123011
.1950347342841

DEGREE OF DENOMINATOR OF ZPTF2   IS  6  (COEFFICIENTS IN ASCENDING ORDER)
0. -46.47353837271 213.7325205421 -416.8323061287 442.3612041218 -256.7878801625
64.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                        CP=  7.23
```

8.9 Example 9 - Multirate Frequency Response by Frequency Decomposition

Problem: Compute the frequency response of the following function

$$G^T(z) = \frac{1}{n}\sum_{k=0}^{n-1} G^{T/n}(z_n e^{j2\pi k/n})$$

where T=.24 seconds, n=3, and $G^{T/n}(z_n)$ is the z transform computed in Example 8.

The above function is the frequency decomposition method for expressing the output transform of a fast-to-slow sampler in terms of the faster input transfer function. An example of how this frequency decomposition method can be applied to the stability analysis of a multirate system is given in Example 19 in Chapter 9.

The operator ZMFRQ evaluates the frequency response of the above function by using only the transform of the faster sampled signal $G^{T/n}(z_n)$. The response is computed by the indicated summation with shifted values of z_n. This operation yields only the multirate frequency response of $G^{T/n}(z_n)$. No z transform at the slower sampling rate is computed. In the next example though, an explicit form for implementing the frequency decomposition method is presented by computing $G^T(s)$ as a rational function .

Since one half of the sampling frequency of the slower output sampler is 13.09 rad/sec, the frequency range to be used for this example will be from 1.0 to 13.0 rad/sec.

The FORTRAN code for this example is:

```
C     EXAMPLE 9
C     TRANSFER FUNCTION IS ZPTF2 FROM PREVIOUS CASE
C
C     ENTER FREQUENCY RESPONSE PARAMETERS FOR ZMRFQ
C
      NOMEGA=2                    "number of values of OMEGA(i)'s to be entered"
      OMEGA(1)=1.                 "OMEGA(1)=first frequency value to be used"
      OMEGA(2)=13.                "OMEGA(NOMEGA)=last frequency value to be used"
      FNICO=1                     ".NE.0 to select Nichols plot"
      FBODE=0                     ".NE.0 to select Bode plot"
      FNYQS=0                     ".NE.0 to select Nyquist plot"
      SAMPT=.24                   "sampling period of slower output sampler"
      NTGER=3                     "integer ratio of output/input sampling periods"
      PRINT*,'ZPTF2 IS AT THE FASTER SAMPLING RATE, SAMPT=.08'
C
      PRINT *,'FREQ. RESP. WILL BE AT THE SLOWER SAMPLING RATE, SAMPT=.2
     +4 '
      TITLE1 =
     +'EXAMPLE 9 MULTIRATE FREQUENCY RESPONSE BY FREQUENCY DECOMPOSITION
     +'
```

```
      CALL ZMRFQ(2)              "compute multirate frequency response of ZPTF2"
C     . . . . . . .
```

The FORTRAN/PRECMP code for this example is:

```
C     EXAMPLE 9
C     TRANSFER FUNCTION IS ZPTF2 FROM PREVIOUS CASE
C
C     ENTER FREQUENCY RESPONSE PARAMETERS FOR ZMRFQ
C
      NOMEGA=2                   "number of values of OMEGA(i)'s to be entered"
*OMEGA 1.  13.  !  OMEGA(1)=FIRST FREQ.,OMEGA(NOMEGA)=LAST FREQ.
      FNICO=1                    ".NE.0 to select Nichols plot"
      FBODE=0                    ".NE.0 to select Bode plot"
      FNYQS=0                    ".NE.0 to select Nyquist plot"
      SAMPT=.24                  "sampling period of slower output sampler"
      PRINT*,'ZPTF2 IS AT THE FASTER SAMPLING RATE, SAMPT=.08'
C     INTEGER RATIO OF OUTPUT/INPUT SAMPLING PERIODS IS ENTERED
C     SECOND ARGUMENT OF ZMRFQ(I,M)
C
      PRINT*,'FREQ. RESP. WILL BE AT THE SLOWER SAMPLING RATE, SAMPT=.2
     +4 '
      TITLE1 =
     +'EXAMPLE 9 MULTIRATE FREQUENCY RESPONSE BY FREQUENCY DECOMPOSITION
     +'
*ZMRFQ 2 3 !  COMPUTE MULTIRATE FREQUENCY RESPONSE OF ZPTF(2)
C     . . . . . . .
```

The printer output for this example is:

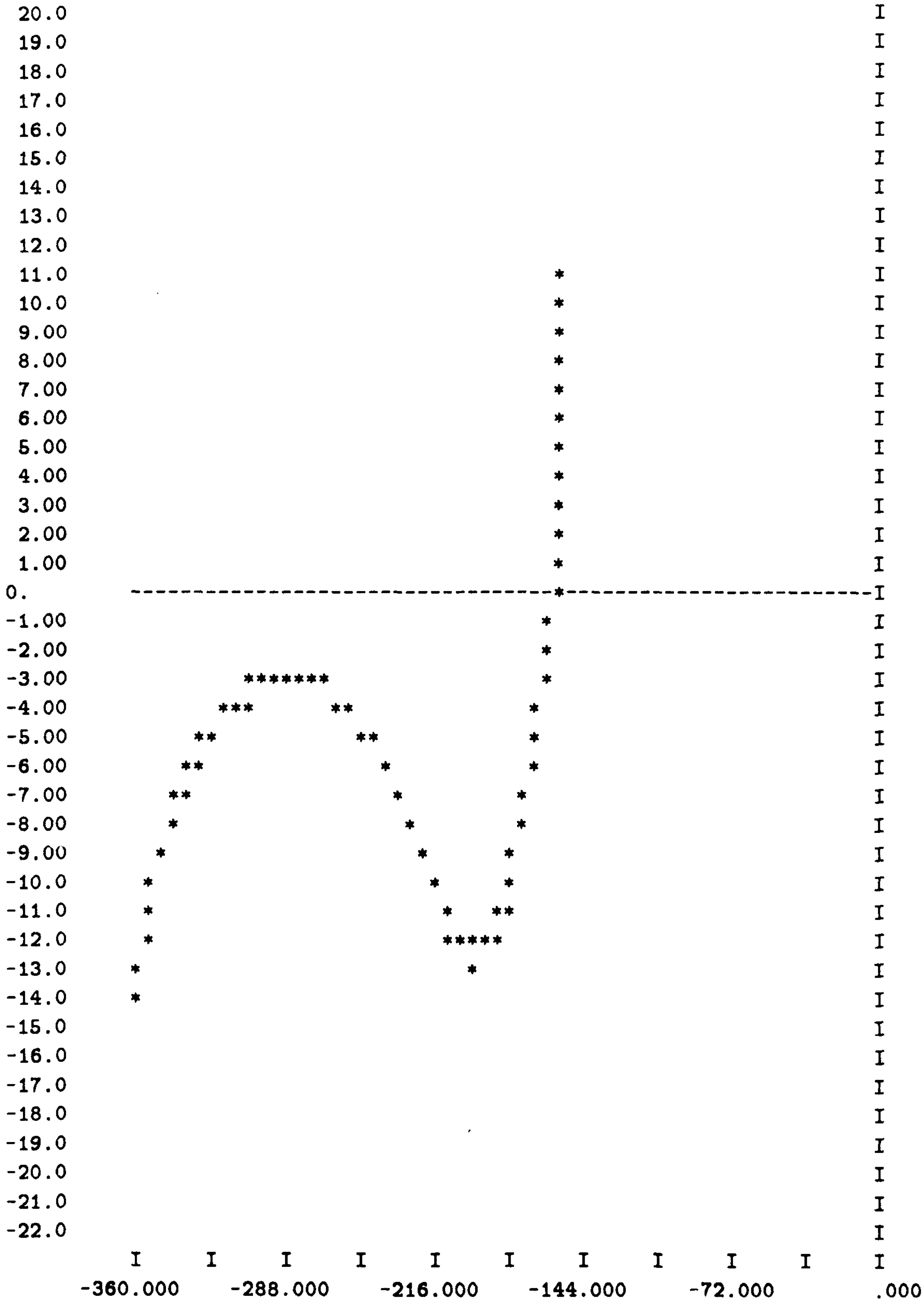
```
ZPTF2 IS AT THE FASTER SAMPLING RATE, SAMPT=.08
FREQ. RESP. WILL BE AT THE SLOWER SAMPLING RATE, SAMPT=.08

                    THE NUMERATOR ROOTS OF ZROOT2   ARE

NO.        REAL          IMAG.           OMEGA          ZETA

 1     -12.268933     0.
 2     -.67135127E-04 0.
 3      .92311634     0.
 4     -.13469859     0.
 5     -1.1567519     0.
```

```
                    LOW ORDER NONZERO COEFFICIENT =   -.23106188E-04

                    THE DENOMINATOR ROOTS OF ZROOT2   ARE

NO.         REAL               IMAG.              OMEGA            ZETA

 1       .61284138          -.73996066         .96078944        -.63785191
 2       .61284138           .73996066         .96078944        -.63785191
 3       .78662786          0.
 4       1.0000000          0.
 5       1.0000000          0.
 6      0.                  0.

                    LOW ORDER NONZERO COEFFICIENT =   -46.473538

DEGREE OF NUMERATOR OF ZPTF2   IS  5    (COEFFICIENTS IN ASCENDING ORDER)
-.00002310618764969 -.3443427037531 -2.507846683739 .679417062974 2.464719123011
.1950347342841

DEGREE OF DENOMINATOR OF ZPTF2   IS  6  (COEFFICIENTS IN ASCENDING ORDER)
0. -46.47353837271 213.7325205421 -416.8323061287 442.3612041218 -256.7878801625
64.

************************************************************
*     ZMRFQ - MULTIRATE FREQUENCY RESPONSE (BY             *
*             FREQUENCY DECOMPOSITION OF ZPTF2             *
************************************************************

SAMPLING PERIOD (SAMPT) = .2400

INTEGER RATIO OF OUTPUT/INPUT SAMPLING PERIODS, NTGER =  3


AUTOMATIC FREQUENCY MODE IF FAUTO.NE.0, FAUTO = 1.000

NOMEGA = 2.000    OMEGA = 1.000    , 13.00    ,

   OMEGA     ZREAL ZIMAG    REAL     IMAGINARY      DB      PHASE   PHASE
  RAD/SEC                                                           MARGIN

  1.000       .997  .080  -.331E+01  -.143E+01   11.125  -156.66    23.34
  1.200       .995  .096  -.247E+01  -.113E+01    8.664  -155.48    24.52
  1.400       .994  .112  -.195E+01  -.906E+00    6.648  -155.08    24.92
  1.700       .991  .136  -.147E+01  -.670E+00    4.171  -155.51    24.49
  2.000       .987  .159  -.117E+01  -.504E+00    2.125  -156.75    23.25
  2.369       .982  .188  -.933E+00  -.360E+00     .000  -158.91    21.09
  2.869       .974  .228  -.722E+00  -.230E+00   -2.411  -162.36    17.64
```

3.469	.962	.274	-.560E+00	-.132E+00	-4.805	-166.75	13.25
4.269	.942	.335	-.424E+00	-.556E-01	-7.377	-172.53	7.47
5.069	.919	.394	-.339E+00	-.112E-01	-9.387	-178.11	1.89
5.346	.910	.415	-.317E+00	.458E-07	-9.966	-180.00	.00
6.946	.850	.528	-.241E+00	.467E-01	-12.206	-190.98	-10.98
8.546	.775	.632	-.225E+00	.989E-01	-12.202	-203.76	-23.76
9.546	.722	.692	-.250E+00	.181E+00	-10.213	-215.89	-35.89
10.05	.694	.720	-.270E+00	.286E+00	-8.096	-226.62	-46.62
10.35	.677	.736	-.261E+00	.409E+00	-6.287	-237.50	-57.50
10.55	.665	.747	-.208E+00	.530E+00	-4.893	-248.60	-68.60
10.70	.656	.755	-.110E+00	.630E+00	-3.887	-260.08	-80.08
10.82	.648	.762	.250E-01	.688E+00	-3.247	-272.08	-92.08
10.92	.642	.767	.160E+00	.690E+00	-2.994	-283.03	-103.03
11.02	.636	.772	.292E+00	.640E+00	-3.054	-294.53	-114.53
11.12	.630	.777	.393E+00	.546E+00	-3.442	-305.70	-125.70
11.22	.623	.782	.447E+00	.435E+00	-4.097	-315.81	-135.81
11.37	.614	.789	.458E+00	.284E+00	-5.376	-328.22	-148.22
11.57	.601	.799	.409E+00	.150E+00	-7.222	-339.91	-159.91
11.82	.585	.811	.335E+00	.654E-01	-9.326	-348.97	-168.97
12.12	.566	.825	.270E+00	.236E-01	-11.343	-355.00	-175.00
12.62	.532	.847	.212E+00	.339E-02	-13.460	-359.09	-179.09
13.00	.506	.862	.198E+00	.333E-03	-14.077	-359.90	-179.90

NICHOLS PLOT (MAGN. VS PHASE) 09/07/88

EXAMPLE 9 MULTIRATE FREQUENCY RESPONSE BY FREQUENCY DECOMPOSITION



The z plane frequencies in the tabulation of the response are at the faster sampling rate. Note that at 13.0 rad/sec the z plane frequency is at (.50g + j.852) or approximately 60 degrees on the unit circle. At the slower sampling rate this point would be at approximately the (-1. + j0.) point, which is 3 times 60 degrees on the unit circle.

8.10 Example 10 - Rational Representation of Frequency Decomposition Method

Problem: Compute the frequency response of the following function

$$G^T(z) = \frac{1}{n}\sum_{k=0}^{n-1} G^{T/n}(z_n e^{j2\pi k/n})$$

by first computing $G^T(z)$ explicitly as a rational function in z. T=.24 seconds, n=3, and $G^{T/n}(z_n)$ is the z transform computed in Example 8.

In Example 9 the frequency response was computed using the command ZMRFQ which numerically evaluated the function over a range of frequencies for z. In this example a different approach is taken. First, $G^T(z)$ is computed as a rational function using the the command ZMRXFM. Then the frequency response of $G^T(z)$ is computed using the command ZFREQ. The method used to compute the rational representation of the frequency decomposition method may be inaccurate for higher order transfer functions. To check the accuracy, the frequency response of the resulting transform can be evaluated and compared with the frequency response computed by the command ZMRFQ. An example of how this frequency decomposition method can be applied to the stability analysis of a multirate system is given in Example 19 of Chapter 9.

The FORTRAN code for this example is:

```
C      EXAMPLE 10
C      G(Z) IS ZPTF2 FROM THE PREVIOUS EXAMPLE
C
C      ENTER PARAMETERS FOR ZMRXFM
C
       SAMPT=.24                 “sampling period of slower output sampler”
       NTGER=3                   “integer ratio of output/input sampling periods”
       PRINT *,'ZPTF2 IS AT THE FASTER SAMPLING RATE, SAMPT=.08'
       PRINT *,'ZPTF3 WILL BE AT THE SLOWER SAMPLING RATE, SAMPT =.24'
       CALL ZMRXFM(3,2)          “compute multirate transform of ZPTF2 and store in SPTF3”
C      . . . . . . . .
C      ZPTF3 IS AT THE SLOWER SAMPLING RATE
C
C      ENTER FREQUENCY RESPONSE PARAMETERS FOR USE WITH ZFREQ
C
       NOMEGA=2                  “number of values of OMEGA to be entered”
       OMEGA(1)=1.               “OMEGA(1)=first frequency value to be used”
       OMEGA(2)=13.              “OMEGA(NOMEGA)=last frequency value to be used”
       TITLE1 =
      +'EXAMPLE 10 RATIONAL REPRESENTATION OF FREQUENCY DECOMPOSITION MET
      +HOD'
       CALL ZFREQ(3)
C      . . . . . . . .
```

The FORTRAN/PRECMP code for this example is:

```
C      EXAMPLE 10
C      G(Z) IS ZPTF2 FROM THE PREVIOUS EXAMPLE
C
C      ENTER PARAMETERS FOR ZMRXFM
C
       SAMPT=.24                “sampling period of slower output sampler”
       NTGER=3                  “integer ratio of output/input sampling periods”
       PRINT *,'ZPTF2 IS AT THE FASTER SAMPLING RATE, SAMPT=.08'
       PRINT *,'ZPTF3 WILL BE AT THE SLOWER SAMPLING RATE, SAMPT =.24'
*ZMRXFM 3 2 !  COMPUTE MULTIRATE TRANSFORM OF ZPTF(2), STORE IN ZPTF(3)
C      . . . . . . .
C      ZPTF3 IS AT THE SLOWER SAMPLING RATE
C
C      ENTER FREQUENCY RESPONSE PARAMETERS FOR USE WITH ZFREQ
C
       NOMEGA=2                 “number of valu  of OMEGA to be entered”
*OMEGA 1 !  OMEGA(1)=FIRST FREQ., OMEGA(NOMEGA)=LAST FREQ.
       TITLE1 =
      +'EXAMPLE 10 RATIONAL REPRESENTATION OF FREQUENCY DECOMPOSITION MET
      +HOD'
*ZFREQ 3 !  COMPUTE FREQUENCY RESPONSE OF ZPTF(3)
C      . . . . . . .
```

The printer output for this example is:

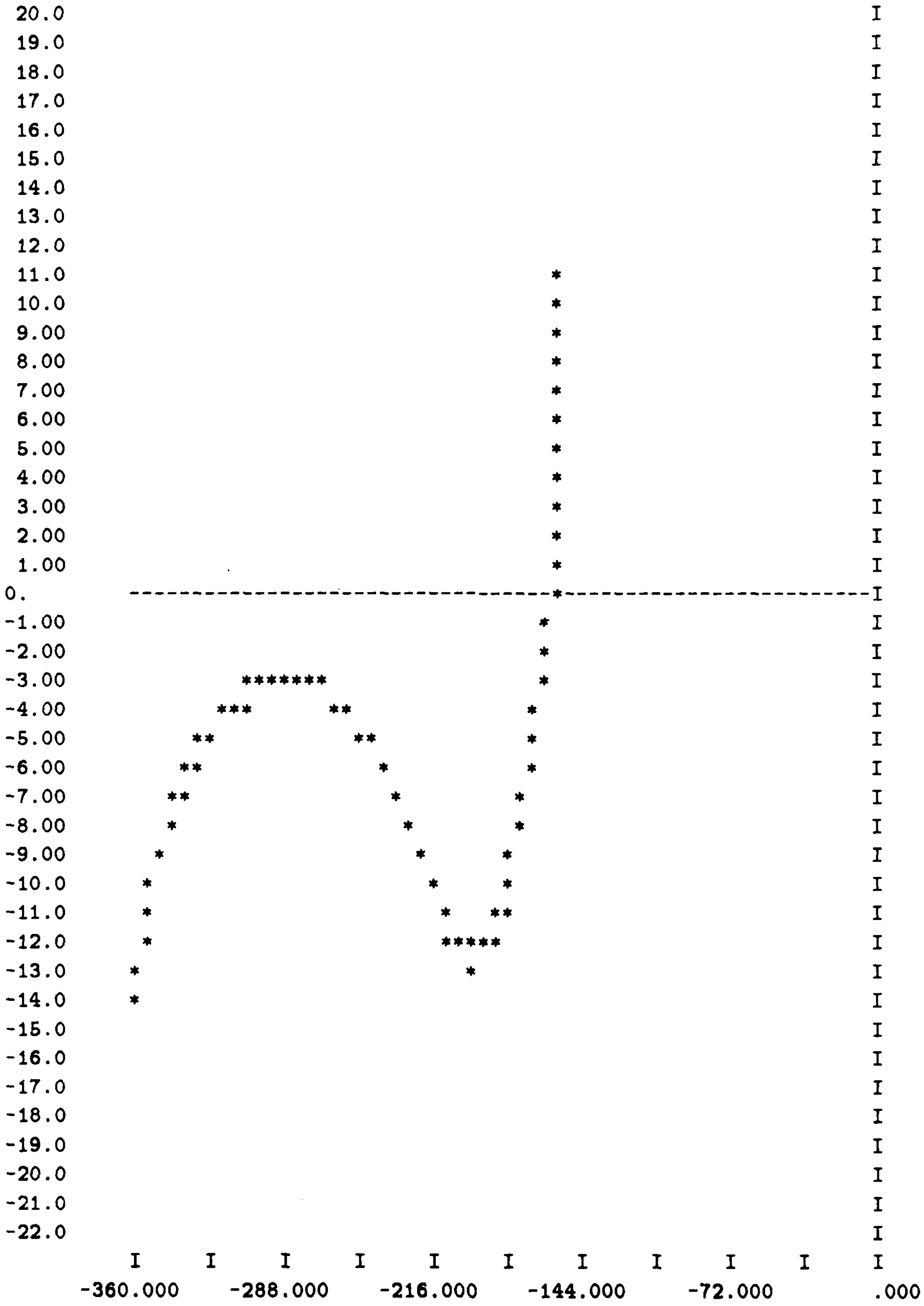
```
ZPTF2 IS AT THE FASTER SAMPLING RATE, SAMPT=.08
ZPTF3 WILL BE AT THE SLOWER SAMPLING RATE, SAMPT=.24

                    THE NUMERATOR ROOTS OF ZROOT2   ARE

NO.        REAL             IMAG.             OMEGA             ZETA

 1     -12.268933       0.
 2     -.67135127E-04   0.
 3      .92311634       0.
 4     -.13469859       0.
 5     -1.1567519       0.

                    LOW ORDER NONZERO COEFFICIENT =   -.23106188E-04
```

```
                        THE DENOMINATOR ROOTS OF ZROOT2   ARE

NO.         REAL              IMAG.              OMEGA             ZETA

 1       .61284138        -.73996066         .96078944         -.63785191
 2       .61284138         .73996066         .96078944         -.63785191
 3       .78662786        0.
 4       1.0000000        0.
 5       1.0000000        0.
 6      0.                0.

                        LOW ORDER NONZERO COEFFICIENT =   -46.473538

DEGREE OF NUMERATOR OF ZPTF2   IS  5    (COEFFICIENTS IN ASCENDING ORDER)
-.00002310618764969 -.3443427037531 -2.507846683739 .679417062974 2.464719123011
.1950347342841

DEGREE OF DENOMINATOR OF ZPTF2   IS  6  (COEFFICIENTS IN ASCENDING ORDER)
0. -46.47353837271 213.7325205421 -416.8323061287 442.3612041218 -256.7878801625
64.

***********************************************
*  ZROOT3  =   ZMRXFM OF        ZROOT2        *
***********************************************

***********************************************
*   ZPTF3  =PSYNTH(ZROOT3  )                  *
***********************************************

                        THE NUMERATOR ROOTS OF ZROOT3   ARE

NO.         REAL              IMAG.              OMEGA             ZETA

 1      -.10647750E-01    0.
 2      -.59161620        0.
 3       .78658382        0.
 4      -2.9655282        0.

                        LOW ORDER NONZERO COEFFICIENT =   -.18162533

                        THE DENOMINATOR ROOTS OF ZROOT3   ARE

NO.         REAL              IMAG.              OMEGA             ZETA

 1      -.77650116        -.42857182         .88692044          .87550262
 2      -.77650116         .42857182         .88692044          .87550262
 3       .48675226        0.
 4       1.0000000        0.
 5      1.0000000       0.

                     LOW ORDER NONZERO COEFFICIENT =   -24.505145

DEGREE OF NUMERATOR OF ZPTF3   IS  4   (COEFFICIENTS IN ASCENDING ORDER)
-.1816253320693 -17.19496629438 -12.53387821719 34.37676333063 12.36036972677

DEGREE OF DENOMINATOR OF ZPTF3   IS  5  (COEFFICIENTS IN ASCENDING ORDER)
-24.50514470241 50.97511989617 39.80519865903 -70.51517819695 -59.75999565584
64.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                               CP=  7.41
************************************************************
*     ZFREQ - FREQUENCY RESPONSE OF Z-PLANE TRANSFER       *
*             FUNCTION 3                                   *
************************************************************

SAMPLING PERIOD (SAMPT) = .2400

AUTOMATIC FREQUENCY MODE IF FAUTO.NE.0, FAUTO = 1.000

NOMEGA = 2.000    OMEGA = 1.000    , 13.00    ,

   OMEGA     ZREAL ZIMAG    REAL     IMAGINARY      DB      PHASE   PHASE
  RAD/SEC                                                           MARGIN
   1.000      .971  .238  -.331E+01  -.143E+01    11.125  -156.66    23.34
   1.200      .959  .284  -.247E+01  -.113E+01     8.664  -155.48    24.52
   1.400      .944  .330  -.195E+01  -.906E+00     6.648  -155.08    24.92
   1.700      .918  .397  -.147E+01  -.670E+00     4.171  -155.51    24.49
   2.000      .887  .462  -.117E+01  -.504E+00     2.125  -156.75    23.25
   2.369      .843  .538  -.933E+00  -.360E+00      .000  -158.91    21.09
   2.869      .772  .635  -.722E+00  -.230E+00    -2.411  -162.36    17.64
   3.469      .673  .740  -.560E+00  -.132E+00    -4.805  -166.75    13.25
   4.269      .519  .854  -.424E+00  -.556E-01    -7.377  -172.53     7.47
   5.069      .347  .938  -.339E+00  -.112E-01    -9.387  -178.11     1.89
   5.346      .284  .959  -.317E+00   .458E-07    -9.966  -180.00      .00
   6.946     -.096  .995  -.241E+00   .467E-01   -12.206  -190.98   -10.98
   8.546     -.462  .887  -.225E+00   .989E-01   -12.202  -203.76   -23.76
   9.546     -.660  .752  -.250E+00   .181E+00   -10.213  -215.89   -35.89
   10.05     -.745  .667  -.270E+00   .286E+00    -8.096  -226.62   -46.62
   10.35     -.791  .612  -.261E+00   .409E+00    -6.287  -237.50   -57.50
   10.55     -.819  .573  -.208E+00   .530E+00    -4.893  -248.60   -68.60
   10.70     -.839  .544  -.110E+00   .630E+00    -3.887  -260.08   -80.08
   10.82     -.855  .518   .250E-01   .688E+00    -3.247  -272.08   -92.08
   10.92     -.868  .497   .160E+00   .690E+00    -2.994  -283.03  -103.03
   11.02     -.879  .476   .292E+00   .640E+00    -3.054  -294.53  -114.53
   11.12     -.890  .455   .393E+00   .546E+00    -3.442  -305.70  -125.70
   11.22     -.901  .434   .447E+00   .435E+00    -4.097  -315.81  -135.81
```

11.37	-.916	.401	.458E+00	.284E+00	-5.376	-328.22	-148.22
11.57	-.934	.357	.409E+00	.150E+00	-7.222	-339.91	-159.91
11.82	-.954	.300	.335E+00	.654E-01	-9.326	-348.97	-168.97
12.12	-.973	.231	.270E+00	.236E-01	-11.343	-355.00	-175.00
12.62	-.994	.112	.212E+00	.339E-02	-13.460	-359.09	-179.09
13.00	-1.000	.022	.198E+00	.333E-03	-14.077	-359.90	-179.90

NICHOLS PLOT (MAGN. VS PHASE) 09/07/88

EXAMPLE 10 RATIONAL REPRESENTATION OF FREQUENCY DECOMPOSITION METHOD



Note that the frequency response is identical to the response computed in Example 9. In the tabulation of the response, the z-plane frequencies are at the slower sampling rate.

8.11 Example 11 - Transfer Function Evaluation by Cramer's Method

Problem: Given

$$\mathbf{M}(s) = \begin{bmatrix} s+2 & -(s+3) \\ 0 & .01s^2 + .15s + 1 \end{bmatrix}$$

$$\mathbf{B}(s) = [0, 1]^T \text{ and } \mathbf{y}(s) = [y_1(s), y_2(s)]^T$$

find the transfer function from u to y_1 by application of Cramer's method.

$$H(s) = \frac{y_1(s)}{u(s)} = \frac{det\ \mathbf{M}_1(s)}{det\ \mathbf{M}(s)}$$

where $\mathbf{M}_1(s)$ is equal to $\mathbf{M}(s)$ with column 1 replaced by $\mathbf{B}(s)$.

Three steps are required for this example. The first two steps compute the determinants and store the results in polynomials and the last step copies the polynomials into an s plane transfer function.

The FORTRAN code for this example is:

```
C      EXAMPLE 11
C      ENTER PARAMETERS FOR DIMENSION AND DEGREE OF MATRIX
C
       MXM=2                        "dimension of matrices"
       MDEG=2                       "highest degree of polynomial elements"
C      ENTER MATRIX DATA FOR COEFFICIENTS OF S**0
       MO(1,1)=2
       MO(1,2)=-3
       MO(2,2)=1
C      ENTER MATRIX DATA FOR COEFFICIENTS OF S**1
       M1(1,1)=1
       M1(1,2)=-1
       M1(2,2)=.15
C      ENTER MATRIX DATA FOR COEFFICIENTS OF S**2
       M2(2,2)=.01
C      ENTER B VECTOR
       BO(2)=1.
C      COMPUTE DETERMINANT OF THE DENOMINATOR OF H(S)
       PRINT *,'DETERMINANT OF DENOMINATOR WILL BE IN POLY2 AND'
       PRINT *,'ROOT2'
```

```
      CALL DTERM(2,0)            "compute determinant of M(s), if second argument
                                  is zero, no column substitution is made"
C     . . . . . . .
C
C     COMPUTE DETERMINANT OF THE NUMERATOR OF H(S)
      PRINT*,'DETERMINANT OF NUMERATOR WILL BE IN POLY3 AND ROOT3'
      CALL DTERM(3,1)            "compute determinant of M1(s), second argument
                                  is column number where B(s) is substituted"
C     . . . . . . .
C
C     COPY POLYNOMIALS TO FORM S PLANE TRANSFER FUNCTION
C
      CALL CPYPS(13,3,2)
C     . . . . . . .
```

The printer output for this example is:

```
DETERMINANT OF DENOMINATOR WILL BE IN POLY2 AND
ROOT2

MATRIX M(S) IS
                      0                 1                 2
ROW  COL             S                 S                 S

 1    1        .20000000E+01    .10000000E+01
 1    2       -.30000000E+01   -.10000000E+01
 2    2        .10000000E+01    .15000000E+00    .10000000E-01

B VECTOR IS
                 0                 1                 2                 3
ROW             S                 S                 S                 S

  2    2        .10000000E+01

************************************************************
*     DTERM - FIND DETERMINANT OF MATRIX                    *
*             WITH NO COLUMN REPLACED BY B VECTOR           *
************************************************************

************************************************************
*    ROOT2   = ROOTS OF DETERMINANT                         *
************************************************************

                      THE ROOTS OF ROOT2   ARE
```

```
NO.        REAL            IMAG.             OMEGA            ZETA

 1     -7.5000000       6.6143783        10.000000        .75000000
 2     -7.5000000      -6.6143783        10.000000        .75000000
 3     -2.0000000      0.

                    LOW ORDER NONZERO COEFFICIENT =    2.0000000

************************************************************
*    POLY2   = COEFFICIENTS OF DETERMINANT POLYNOMIAL      *
************************************************************

DEGREE OF POLY2   IS  3           (COEFFICIENTS IN ASCENDING ORDER)
2. 1.3 .17 .01

DETERMINANT OF NUMERATOR WILL BE IN POLY3 AND ROOT3

MATRIX M(S) IS
                     0                1                2
ROW  COL            S                S                S

 1    1       .20000000E+01    .10000000E+01
 1    2      -.30000000E+01   -.10000000E+01
 2    2       .10000000E+01    .15000000E+00    .10000000E-01

B VECTOR IS
                0                1                2                3
ROW            S                S                S                S

  2    2       .10000000E+01

************************************************************
*     DTERM - FIND DETERMINANT OF MATRIX                   *
*             WITH COLUMN  1 REPLACED BY B VECTOR          *
************************************************************

************************************************************
*    ROOT3   = ROOTS OF DETERMINANT                        *
************************************************************

                           THE ROOTS OF ROOT3   ARE

NO.        REAL            IMAG.             OMEGA            ZETA

 1     -3.0000000      0.

                    LOW ORDER NONZERO COEFFICIENT =    3.0000000
```

```
*************************************************************
*    POLY3   = COEFFICIENTS OF DETERMINANT POLYNOMIAL      *
*************************************************************

DEGREE OF POLY3   IS  1           (COEFFICIENTS IN ASCENDING ORDER)
3. 1.

                             THE ROOTS OF ROOT3   ARE

NO.        REAL            IMAG.            OMEGA            ZETA

 1     -3.0000000       0.

                      LOW ORDER NONZERO COEFFICIENT =    3.0000000

DEGREE OF POLY3   IS  1           (COEFFICIENTS IN ASCENDING ORDER)
3. 1.

                             THE ROOTS OF ROOT2   ARE

NO.        REAL            IMAG.            OMEGA            ZETA

 1     -7.5000000       6.6143783        10.000000        .75000000
 2     -7.5000000      -6.6143783        10.000000        .75000000
 3     -2.0000000       0.

                      LOW ORDER NONZERO COEFFICIENT =    2.0000000

DEGREE OF POLY2   IS  3           (COEFFICIENTS IN ASCENDING ORDER)
2. 1.3 .17 .01

************************************************
*  SROOT13 =       ROOT3     /      ROOT2      *
************************************************

************************************************
*   SPTF13 =       POLY3     /      POLY2      *
************************************************

                      THE NUMERATOR ROOTS OF SROOT13  ARE

NO.        REAL            IMAG.            OMEGA            ZETA

 1     -3.0000000       0.

                      LOW ORDER NONZERO COEFFICIENT =    3.0000000
```

```
                    THE DENOMINATOR ROOTS OF SROOT13  ARE

NO.        REAL             IMAG.            OMEGA             ZETA

 1      -7.5000000        6.6143783        10.000000        .75000000
 2      -7.5000000       -6.6143783        10.000000        .75000000
 3      -2.0000000       0.

                    LOW ORDER NONZERO COEFFICIENT =    2.0000000

DEGREE OF NUMERATOR OF SPTF13  IS  1    (COEFFICIENTS IN ASCENDING ORDER)
3. 1.

DEGREE OF DENOMINATOR OF SPTF13  IS  3  (COEFFICIENTS IN ASCENDING ORDER)
2. 1.3 .17 .01

BODE GAIN =    1.5000000
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                            CP=  7.55
```

8.12 Example 12 - Save Data From Batch Job

Problem: Save all **POLY**$_i$, **SPTF**$_i$, **ZPTF**$_i$, **WPTF**$_i$, and matrix data (used by DTERM and DETRM) so that it can be used in a subsequent batch or interactive LCAP2 job. Assume that Examples 1 through 11 were all run on the same batch job.

The command SAVE will save all pertinent data (except for the transfer connection data[1]) from a current job so that on subsequent batch or interactive jobs this data can be reloaded with the LOAD command. The data will be written to a local file specified by the first argument of the SAVE command. Upon completion of the batch job this file must be SAVEed on the CRAY or CATALOGed on the CDC.

To label the data to be stored, the contents of the character variable STRLBL in common block HEAD is written to the file.

The FORTRAN code for this example is:

```
C      EXAMPLE 12
C
C      SUBROUTINE SAVE BELOW WILL SAVE ALL TRANSFER FUNCTIONS,
C      POLYNOMIALS, AND MATRIX ELEMENTS INTO A LOCAL FILE SPECIFIED
C      BY THE FIRST ARGUMENT OF THE SUBROUTINE. A NON ZERO VALUE.
C      FOR THE SECOND ARGUMENT WILL PRINT OUT ALL THE DATA SAVED.
```

[1] See Reference for commands B1SAVE and B2SAVE.

```
C      THE CONTENTS OF THE CHARACTER VARIABLE SAVLBL (IN COMMON/HEAD/)
C      WILL BE SAVE IN THE FILE AND BE PRINTED OUT WHEN THE FILE IS
C      LOADED IN A SUBSEQUENT JOB.
C
       SAVLBL =
     + 'EXAMPLES FOR BATCH LCAP2 USERS MANUAL '
C
       CALL SAVE('SAVDATA',1)      “first argument is the file name
                                    if second argument is 0 printout will be suppressed”
```

The printer output for this example is[2]:

```
************************************************************
*     SAVE - SAVE ALL CURRENT DATA (POLYNOMIAL, TRANSFER   *
*            FUNCTION, MATRIX, LCAP2 PARAMETERS), EXCEPT   *
*            FOR TRANSFER FUNCTION CONNECTION DATA         *
************************************************************

LABEL FOR FILE STRDATA IS
EXAMPLES FOR BATCH LCAP2 USERS MANUAL

DEGREE OF NUMERATOR OF SPTF1   IS  0   (COEFFICIENTS IN ASCENDING ORDER)
25.

DEGREE OF DENOMINATOR OF SPTF1   IS  2  (COEFFICIENTS IN ASCENDING ORDER)
25. 5. 1.

BODE GAIN =    1.0000000

                    THE NUMERATOR ROOTS OF SROOT2   ARE

NO.        REAL            IMAG.             OMEGA            ZETA

 1     -10.000000       0.

                    LOW ORDER NONZERO COEFFICIENT =    30.000000

                    THE DENOMINATOR ROOTS OF SROOT2   ARE

NO.        REAL            IMAG.             OMEGA            ZETA
```

[2] Since the output from Examples 1 through 11 has been listed already, only portions of the output generated by this example will be shown.

```
1     -1.0000000       2.0000000       2.2360680       .44721360
2     -1.0000000      -2.0000000       2.2360680       .44721360
3     -7.0000000      0.
4     0.              0.

                     LOW ORDER NONZERO COEFFICIENT =    70.000000

DEGREE OF NUMERATOR OF SPTF2   IS  1    (COEFFICIENTS IN ASCENDING ORDER)
30. 3.

DEGREE OF DENOMINATOR OF SPTF2   IS  4  (COEFFICIENTS IN ASCENDING ORDER)
0. 70. 38. 18. 2.

BODE GAIN =     .42857143
```

⋮

(data for $\mathbf{SPTF}_i, i = 3, ...$)

⋮

(data for $\mathbf{WPTF}_i, i = 1, 2, ...$)

⋮

(data for $\mathbf{WPTF}_i, i = 1, 2, ...$)

⋮

(data for $\mathbf{POLY}_i, i = 1, 2, ...$)

⋮

```
MATRIX M(S) IS
                     0                1                2
ROW  COL            S                S                S

 1    1       .20000000E+01    .10000000E+01
 1    2      -.30000000E+01   -.10000000E+01
 2    2       .10000000E+01    .15000000E+00    .10000000E-01

B VECTOR IS
                0                 1                2                 3
ROW            S                 S                S                 S

  2   2       .10000000E+01
```

Upon exit from batch LCAP2, the local file name "SAVDATA" created by the SAVE command must be (1) SAVEd if the CRAY is used or (2) CATALOGed if the CDC is used. One of the following statements is to be used.

```
SAVE,DN=SAVDATA,PDN=CBUSERDATA,OWN=9487.     "for CRAY"
CATALOG,SAVDATA,8BBUSERDATA,ID=9487.         "FOR MFB CDC"
CATALOG,SAVDATA,8XBUSERDATA,ID=9487,ST=PF6. "for MFX CDC, (file is saved on MFB)"
```

8.13 Example 13 - Load Data Saved From Example 12

Problem: Load data saved from Example 12

The command LOAD will load $\mathbf{POLY}_i$, $\mathbf{SPTF}_i$, $\mathbf{ZPTF}_i$, $\mathbf{WPTF}_i$, and matrix data from a local file. Before using this command, the local file must be (1) ACCESSed (CRAY) or (2) ATTACHed (CDC) before batch LCAP2 is executed. This is done by one of the following statements:

```
ACCESS,DN=OLDDATA,PDN=CBUSERDATA,OWN=9487. "for CRAY"
ATTACH(OLDDATA,8BBUSERDATA,ID=9487)        "for MFB CDC"
ATTACH(OLDDATA,8XBUSERDATA,ID=9487,ST=PF6) "for MFX CDC"
```

The FORTRAN code for this example is:

```
C     LOAD DATA SAVED IN EXAMPLE 12
C
      CALL LOAD('OLDDATA',1)       "first argument is the file name"
                                   "if second argument is 0 printout will be suppressed"
```

The printer output for this example is[1]:

```
***********************************************************
*     LOAD - LOAD POLYNOMIAL, TRANSFER FUNCTION, MATRIX,  *
*            LCAP2 PARAMETERS (BUT NOT TRANSFER           *
*            FUNCTION CONNECTION DATA) FROM A FILE        *
***********************************************************

LABEL FOR FILE STRDATA IS
EXAMPLES FOR BATCH LCAP2 USERS MANUAL

DEGREE OF NUMERATOR OF SPTF1   IS  0    (COEFFICIENTS IN ASCENDING ORDER)
25.
```

[1] Only the first part of the printout is shown. Note that the labeling information entered with the character variable SAVLBL in Example 12 is printed out.

```
DEGREE OF DENOMINATOR OF SPTF1   IS  2  (COEFFICIENTS IN ASCENDING ORDER)
25. 5. 1.

BODE GAIN =     1.0000000

                      THE NUMERATOR ROOTS OF SROOT2   ARE

NO.         REAL              IMAG.              OMEGA            ZETA

 1      -10.000000       0.

                      LOW ORDER NONZERO COEFFICIENT =    30.000000

                      THE DENOMINATOR ROOTS OF SROOT2   ARE

NO.         REAL              IMAG.              OMEGA            ZETA

 1      -1.0000000        2.0000000        2.2360680        .44721360
 2      -1.0000000       -2.0000000        2.2360680        .44721360
 3      -7.0000000       0.
 4      0.               0.

                      LOW ORDER NONZERO COEFFICIENT =    70.000000

                                    ⋮
```

Chapter 9

Advanced Examples

Examples are presented in this chapter to demonstrate some of the different methods available in LCAP2 for solving complex problems. Examples 14 and 15 describe two related methods which can be used to model a complex continuous system represented by a set of Laplace transformed differential equations. Most of the stability analysis for launch vehicles at Aerospace have been formulated using these two methods. Example 16 describes two methods for computing the frequency response of a nonrational function. The use of a user defined function is presented for the generalized s plane frequency response command, FREQS.

Example 17 is the solution of the IEEE CACSD Benchmark [4] simple continuous model using the automated method based on connection of transfer function blocks. Examples 18 through 21 are solutions of the IEEE CACSD Benchmark Problem No. 3 [5] multi-rate sampled data autopilots. Example 18 is the solution of the 2-rate model using the automated method based on connection of transfer function blocks while Example 19 is the same solution using classical transform methods. Examples 20 and 21 are analogous solutions for the 4-rate model. The complete setup for these two problems is given in the examples. However, since the output for these two examples will be very long, only selected output will be given in this manual. While only a single printout of a set of transfer function roots is sufficient to explain the output of a B1TF or B2TF command, the complete solution for these benchmark problems will be presented since these examples will also serve to benchmark the LCAP2 program.

The use of classical transform methods in Examples 19 and 21 are presented as an alternate method for solving the multirate problems. Unlike the method used in Example 18 and 20, which is completely automated so that the analyst with very little experience in multirate theory can easily set up an analysis, the classical transform method is more involved and requires a basic understanding of multirate relationships in the frequency domain. Example 19 presents a review of two basic z plane multirate relationships and shows how the SZXFM and ZMRXFM commands are used. This block diagram reduction method provides some insight to the problem not offered by the automated method of analysis which is based on the Kalman-Bertram state space method. In Example 21 the same type of multirate operations are used, except that, for computional reasons, the analysis is performed in the w plane instead.

Examples 17 through 21 can be reproduced by the user by executing any of the following files which are saved as indirect files on CDC MFB. A copy of an indirect file can be obtained by typing the following command in INTERCOM:

```
IGET,file_name/PF=ZL2USER,ID=9487
```

where **file_name** and its description is given in Table 9.1.

Table 9.1: Decks for Creation of Examples 17-21

Files For Reproducing Examples 16 - 19	
file_name	Description
CEX17	For CRAY, Example 17
CEX18	For CRAY, Example 18
CEX19	For CRAY, Example 19
CEX20	For CRAY, Example 20
CEX21	For CRAY., Example 21
BEX17	For CDC MFB, Example 17
BEX18	For CDC MFB, Example 18
BEX19	For CDC MFB, Example 19
BEX20	For CDC MFB, Example 20
XEX17	For CDC MFX, Example 17
XEX18	For CDC MFX, Example 18
XEX19	For CDC MFX, Example 19
XEX20	For CDC MFX, Example 20

To execute a CRAY job, type:

```
CSUBMIT,file_name.
```

To execute a CDC job, type:

```
BATCH,file_name,INPUT.
```

9.1 Example 14 - Modeling a System as a Matrix of Laplace Transformed Differential Equations

Problem: Model the following system as a matrix of Laplace transformed differential equations and then compute the closed loop transfer function θ/u and the open loop transfer function v/u by using Cramer's method.

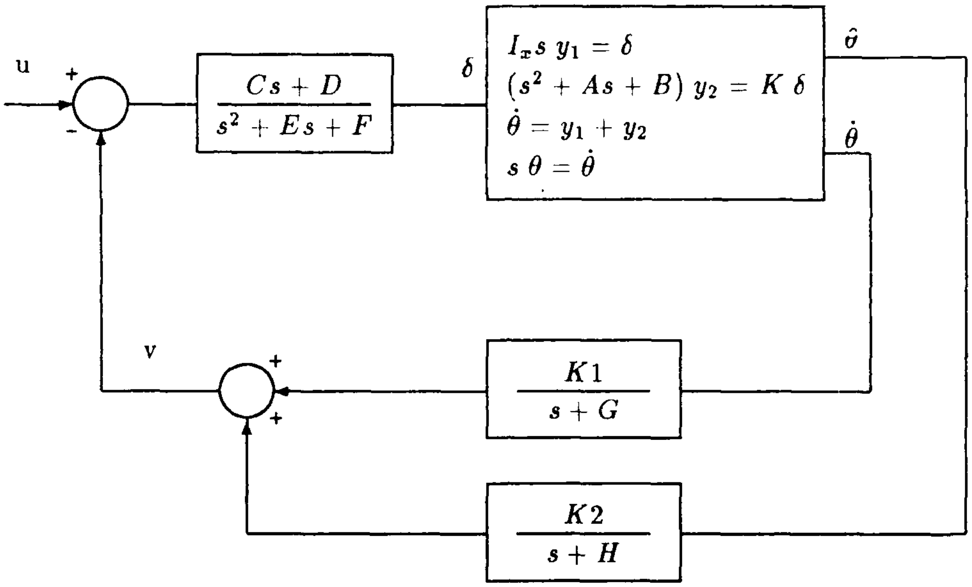


Figure 9.1: Continuous System with Plant Represented by a Set of Equations

This example illustrates a method of analysis for a continuous system which is not completely modeled as a connection of transfer function blocks. It is representative of complex systems such as launch vehicles in which the explicit evaluation of plant transfer functions (i.e, $\dot{\theta}/\delta$ and θ/δ for this example) is computationally not the best procedure to use[1] for stability analysis. If plant transfer functions were computed so that either the (1) block diagram reduction method or the (2) automated analysis method using transfer function connection blocks was used to evaluate the desired open and closed loop transfer functions, the common denominator roots of the plant transfer functions will introduce redundant states (roots) to the analysis model of the system. When both the number of plant outputs and the order of the plant transfer functions are small, the number of redundant roots will be small and will generally not present a problem for LCAP2. However, if there is more than one plant output and the order of the plant transfer functions is not small, the number of redundant roots will be excessive. Computationally, the additional redundant roots will require more execution time and may lead to less accurate results. Also if the number of redundant roots is very large, the size of the model may be too large for LCAP2 to analyze. For this type of problem, the system is best modeled by a set of Laplace transformed differential equations.

Although the plant for this example is low order and transfer functions $\dot{\theta}/\delta$ and θ/δ can be computed explicitly so that the block diagram reduction method can be used instead, this low order example is used only to demonstrate the techniques of this matrix method of analysis.

[1] Except for checking out the modeling of the plant dynamics.

For stability analysis the loop is broken after the summation of the feedback loop. Define a flag KFLG, whose value is to be 0 or 1, which will be used to open or close this feedback loop. The above figure is redrawn with this flag to obtain Figure 9.2.

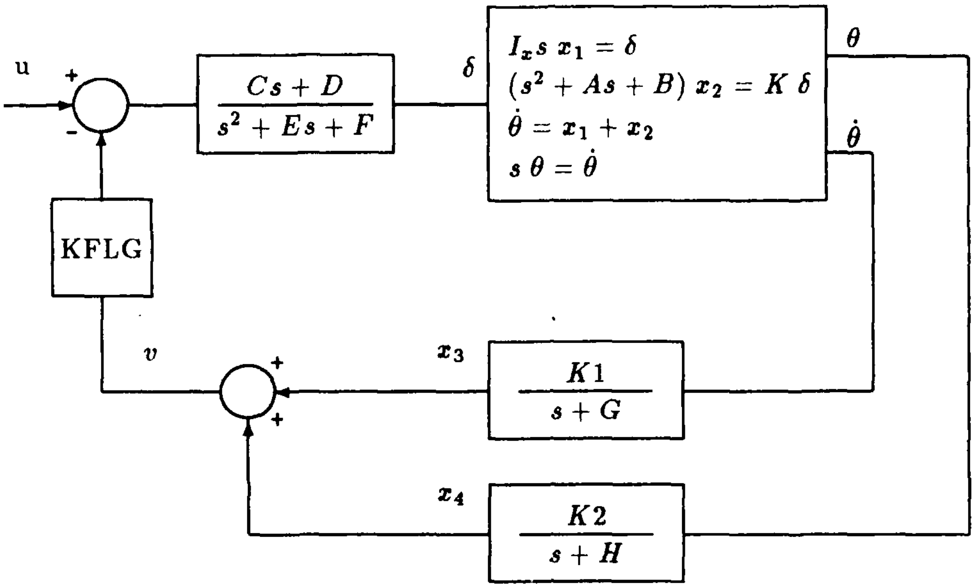


Figure 9.2: Figure 9.1 Redrawn with KFLG

The set of Laplace transformed equations for this figure is:

$$\begin{aligned}
(s^2+Es+F)\delta &= (Cs+D)(u - KFLG * v) \\
(I_x s)x_1 &= \delta \\
(s^2+As+B)x_2 &= K\delta \\
\dot{\theta} &= x_1 + x_2 \\
s\theta &= \dot{\theta} \\
(s+G)x_3 &= K_1\dot{\theta} \\
(s+H)x_4 &= K_2\theta \\
v &= x_3 + x_4
\end{aligned}$$

The matrix form of these equations is

$$\mathbf{M}(s)\mathbf{y}(s) = \mathbf{B}(s)u$$

where $\mathbf{y}(s)$, $\mathbf{M}(s)$, and $\mathbf{B}(s)$ are given by

$$\mathbf{y}(s) = \left[\delta, x_1, x_2, \dot{\theta}, \theta, x_3, x_4, v\right]^T$$

$$\mathbf{M}(s) = \begin{bmatrix} s^2+Es+F & 0 & 0 & 0 & 0 & 0 & 0 & KFLG*(Cs+D) \\ -1 & I_x s & 0 & 0 & 0 & 0 & 0 & 0 \\ -K & 0 & s^2+As+B & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -K_1 & 0 & s+G & 0 & 0 \\ 0 & 0 & 0 & 0 & -K_2 & 0 & s+H & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 \end{bmatrix}$$

$$\mathbf{B}(s) = [(Cs+D),0,0,0,0,0,0,0]^T$$

The closed loop transfer function $\theta(s)/u(s)$ is given by

$$\frac{\theta(s)}{u(s)} = \frac{det\ \mathbf{M}_5(s)}{det\ \mathbf{M}(s)}$$

where $\mathbf{M}_5(s)$[1] is defined to be equal to $\mathbf{M}(s)$ with column 5 replaced by $\mathbf{B}(s)$ and the value of KFLG is set to 1.

The open loop transfer function $v(s)/u(s)$ is given by

$$\frac{v(s)}{u(s)} = \frac{det\ \mathbf{M}_8(s)}{det\ \mathbf{M}(s)}$$

where the value of KFLG set to 0.

To use LCAP2 to evaluate the above transfer functions, the polynomial matrix M(s) is represented by the input matrices M0, M1, M2, M3, and M4 as

$$\mathbf{M}(s) = M4\ s^4 + M3\ s^3 + M2\ s^2 + M1\ s + M0$$

and the input vector $\mathbf{B}(s)$ is represented by the input vectors B0, B1, B2, B3, and B4 as

$$\mathbf{B}(s) = B4\ s^4 + B3\ s^3 + B2\ s^2 + B1\ s + B0$$

The FORTRAN code for this example is:

```
C      ENTER PARAMETERS FOR ORDER AND DEGREE OF MATRIX
C
       MXM=8                         “order of matrix”
       MDEG=2                        “highest degree of polynomial element”
```

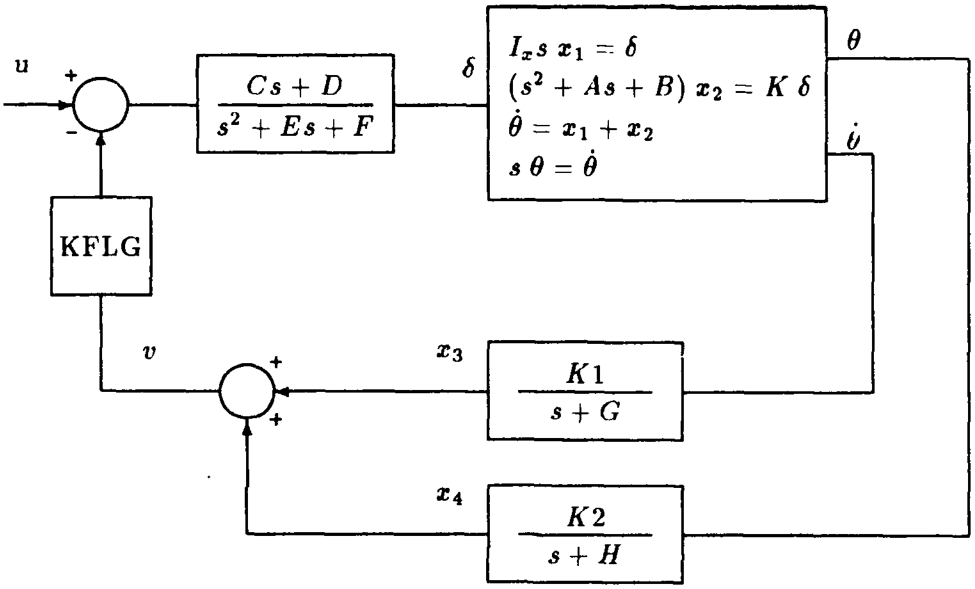
[1] By definition, $\mathbf{M}_0(s) = \mathbf{M}(s)$

```
C
C      ENTER DATA
       A=.....
       B=.....
       C=.....
       D=.....
       E=.....
       F=.....
       G=.....
       H=.....
       RK=.....                  “RK is $k$”
       RK1=.....                 “RK1 is $K_1$”
       RK2=.....                 “RK2 is $K_2$”
       RIX=.....                 “RIX is $I_x$”
C      ENTER DATA FOR CLOSED LOOP CASE FIRST
       MO(1,1)=F
       MO(1,8)=D                 “for KFLG=1”
       MO(2,1)=-1.
       MO(3,1)=-RK
       MO(3,3)=B
       MO(4,2)=-1.
       MO(4,3)=-1.
       MO(4,4)=1.
       MO(5,4)=-1.
       MO(5,5)=1.
       MO(6,4)=-RK1
       MO(6,6)=G
       MO(7,5)=-RK2
       MO(7,7)=H
       MO(8,6)=-1.
       MO(8,7)=-1.
       MO(8,8)=1
       M1(1,1)=E
       M1(1,8)=C                 “for KFLG=1”
       M1(2,2)=RIX
       M1(3,3)=A
       M1(6,6)=G
       M1(7,7)=H
       M2(1,1)=1.
       BO(1)=D
       B1(1)=C
C      COMPUTE DENOMINATOR FOR CLOSED LOOP
C
       CALL DTERM(1,0)           “determinant of M(s) stored in $\mathbf{POLY}_1$”
C
C      COMPUTE NUMERATOR FOR CLOSED LOOP
C
       CALL DTERM(2,5)           “determinant of $M_5$(s) stored in $\mathbf{POLY}_2$”
```

```
C
C     COPY NUM. AND DENOM. INTO CLOSED LOOP TRANSFER FUNCTION
C
      CALL CPYPS(1,2,1)          "$SPTF_1$ = $POLY_2$ / $POLY_1$"
C
C     CHANGE ELEMENTS FOR OPEN LOOP CONFIGURATION
C
      M0(1,8)=0.                 "since KFLG=0"
      M1(1,8)=0.                 "since KFLG=0"
C
C     COMPUTE DENOMINATOR FOR OPEN LOOP
C
      CALL DTERM(3,0)            "determinant of M(s) stored in $POLY_3$"
C
C     COMPUTE NUMERATOR FOR OPEN LOOP
C
      CALL DTERM(4,8)            "determinant of $M_8$(s) stored in $POLY_4$"
C
C     COPY NUM. AND DENOM. INTO OPEN LOOP TRANSFER FUNCTION
C
      CALL CPYPS(2,4,3)          "$SPTF_2$ = $POLY_4$ / $POLY_3$"
```

9.2 Example 15 - Modeling a System as a Matrix of Laplace Transformed Differential Equations Using the Augmented Method

Problem: Model the following system as a matrix of Laplace transformed differential equations using the augmented method and then compute the closed loop transfer function θ/u and the open loop transfer function v/u by using Cramer's method.



u
+
-
$\frac{Cs + D}{s^2 + Es + F}$
δ
$I_x s\, x_1 = \delta$
$(s^2 + As + B)\, x_2 = K\, \delta$
$\dot{\theta} = x_1 + x_2$
$s\, \theta = \dot{\theta}$
θ
$\dot{\theta}$
KFLG
v
+
+
x_3
$\frac{K1}{s + G}$
x_4
$\frac{K2}{s + H}$

Figure 9.3: Continuous System with Plant Represented by a Set of Equations

This example is similar to Example 14 except that the augmented method is used. In that example, using Cramer's method, a transfer function for a system represented by

$$\mathbf{M}(s)\, \mathbf{y}(s) = \mathbf{B}(s)\, u(s)$$

was given as:

$$\frac{y_j(s)}{u(s)} = \frac{\det\, \mathbf{M}_j(s)}{\det\, \mathbf{M}(s)}$$

where $\mathbf{M}_j$(s) is equal to **M**(s) with column j replaced by **B**(s).

Before the DTERM command was implemented in LCAP2, the user had to manually substitute the **B**(s) vector into the j-th column of matrix **M**(s) in order to compute the numerator of a transfer function. If more than one transfer function were to be computed, additional effort was required by the user to restore the original column of **M**(s) before another determinant was computed. The augmented method is a means of simplifying this operation so that the setup effort for computing different numerators and denominators involves only changing a single element of the matrix.

If the order of the system matrix **A**(s) is nxn, augment the vector **y** by the input u so that the system can be represented as

$$\mathbf{N}(s) = \begin{bmatrix} \mathbf{M}(s) & -\mathbf{B}(s) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{y}(s) \\ u(s) \end{bmatrix} = 0$$

Define $\mathbf{N}(\mathrm{s})_k$ as a (n+1)x(n+1) matrix whose first n rows are equal to $\mathbf{N}(\mathrm{s})$ and all elements of row n+1 are equal to 0 except for column k which is equal to 1. Then, by Cramer's method, the transfer function between the input u and the x_j is given by

$$\frac{\mathrm{y}_j(\mathrm{s})}{\mathrm{u}(\mathrm{s})} = \frac{\det \mathbf{N}_j(\mathrm{s})}{\det \mathbf{N}_{n+1}(\mathrm{s})}$$

Even though the DTERM command used in Example 14 for solving this problem automatically inserts the $\mathbf{B}(\mathrm{s})$ vector into the system matrix for the user, the augmented matrix is still being used. It is used extensively in the stability analysis programs for launch vehicles since many of those analysis programs were implemented before the DTERM command was available. This example is intended to illustrate the utility of this method of modeling a complex continuous system.

The set of Laplace transformed equations for this problem is:

$$\begin{aligned} (s^2 + Es + F)\delta &= (Cs + D)(u - KFLG * v) \\ (I_x s)x_1 &= \delta \\ (s^2 + As + B)x_2 &= K\delta \\ \dot{\theta} &= x_1 + x_2 \\ s\theta &= \dot{\theta} \\ (s + G)x_3 &= K_1\dot{\theta} \\ (s + H)x_4 &= K_2\theta \\ v &= x_3 + x_4 \end{aligned}$$

The augmented matrix form of these equations is

$$\begin{bmatrix} \mathbf{M}(s) & -\mathbf{B}(s) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{y}(s) \\ u(s) \end{bmatrix} = 0$$

where $\mathbf{x}(s)$, $\mathbf{M}(s)$, and $\mathbf{B}(s)$ are given by

$$\mathbf{x}(s) = \left[\delta, x_1, x_2, \dot{\theta}, \theta, x_3, x_4, v\right]^T$$

$$\mathbf{M}(s) = \begin{bmatrix} s^2 + Es + F & 0 & 0 & 0 & 0 & 0 & 0 & KFLG * (Cs + D) \\ -1 & I_x s & 0 & 0 & 0 & 0 & 0 & 0 \\ -K & 0 & s^2 + As + B & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -K_1 & 0 & s + G & 0 & 0 \\ 0 & 0 & 0 & 0 & -K_2 & 0 & s + H & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 \end{bmatrix}$$

$$\mathbf{B}(s) = [(Cs + D), 0, 0, 0, 0, 0, 0, 0]^T$$

The closed loop transfer function $\theta(s)/u(s)$ is given by

$$\frac{\theta(s)}{u(s)} = \frac{det\ \mathbf{N}_5(s)}{det\ \mathbf{N}_9(s)}$$

where

$$\mathbf{N}_5(s) = \begin{bmatrix} \mathbf{M}(s) & -\mathbf{B}(s) \\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0 & 0 \end{bmatrix}$$

and

$$\mathbf{N}_9(s) = \begin{bmatrix} \mathbf{M}(s) & -\mathbf{B}(s) \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 & 1 \end{bmatrix}$$

Similarly, the open loop transfer function $v(s)/u(s)$ is given by

$$\frac{v(s)}{u(s)} = \frac{det\ \mathbf{N}_8(s)}{det\ \mathbf{N}_9(s)}$$

where the value of KFLG is set to 0.

To use LCAP2 to evaluate the above transfer functions, the polynomial matrix $\mathbf{N}_k$(s) is represented by the input matrices M0, M1, M2, M3, and M4 as

$$\mathbf{N}_k(s) = M4\ s^4 + M3\ s^3 + M2\ s^2 + M1\ s + M0$$

The FORTRAN code for this example is:

```
C      ENTER PARAMETERS FOR ORDER AND DEGREE OF MATRIX
C
       MXM=9                        "order of (n+1)x(n+1) matrix"
       MDEG=2                       "highest degree of polynomial element"
C
C      ENTER DATA
       A=.....
       B=.....
       C=.....
       D=.....
       E=.....
       F=.....
```

```
      G=.....
      H=.....
      RK=.....                 "RK is $k$"
      RK1=.....                "RK1 is $K_1$"
      RK2=.....                "RK2 is $K_2$"
      RIX=.....                "RIX is $I_x$"
C     ENTER DATA FOR CLOSED LOOP CASE FIRST
      MO(1,1)=F
      MO(1,8)=D                "for KFLG=1"
      MO(1,9)=-D               "for s**0 term of B(s)"
      MO(2,1)=-1.
      MO(3,1)=-RK
      MO(3,3)=B
      MO(4,2)=-1.
      MO(4,3)=-1.
      MO(4,4)=1.
      MO(5,4)=-1.
      MO(5,5)=1.
      MO(6,4)=-RK1
      MO(6,6)=G
      MO(7,5)=-RK2
      MO(7,7)=H
      MO(8,6)=-1.
      MO(8,7)=-1.
      MO(8,8)=1
      MO(9,9)=1
      M1(1,1)=E
      M1(1,8)=C                "for KFLG=1"
      M1(1,9)=-C               "for s**1 term of B(s)"
      M1(2,2)=RIX
      M1(3,3)=A
      M1(6,6)=G
      M1(7,7)=H
      M2(1.1)=1.
C     COMPUTE DENOMINATOR FOR CLOSED LOOP
C
      CALL DETRM(1)            "determinant of $N_9(s)$ stored in $POLY_1$"
C
C     CHANGE ELEMENTS FOR NUMERATOR
C
      MO(9,9)=0.
      MO(9,5)=1.
C
C     COMPUTE NUMERATOR FOR CLOSED LOOP
C
      CALL DETRM(2)            "determinant of $N_5(s)$ stored in $POLY_2$"
C
C     COPY NUM. AND DENOM. INTO CLOSED LOOP TRANSFER FUNCTION
```

```
C
      CALL CPYPS(1,2,1)          "SPTF1 = POLY2 / POLY1"
C
C     CHANGE ELEMENTS FOR OPEN LOOP CONFIGURATION
C
      MO(1,8)=0.                 "since KFLG=0"
      M1(1,8)=0.                 "since KFLG=0"
C
C     CHANGE ELEMENTS FOR DENOMINATOR
C
      MO(9,5)=0.
      MO(9,9)=1.
C
C     COMPUTE DENOMINATOR FOR OPEN LOOP
C
      CALL DETRM(3)              "determinant of N9(s) stored in POLY3"
C
C     CHANGE ELEMENTS FOR NUMERATOR
C
      MO(9,9)=0.
      MO(9,8)=1.
C
C     COMPUTE NUMERATOR FOR OPEN LOOP
C
      CALL DETRM(4)              "determinant of N8(s) stored in POLY4"
C
C     COPY NUM. AND DENOM. INTO OPEN LOOP TRANSFER FUNCTION
C
      CALL CPYPS(2,4,3)          "SPTF2 = POLY4 / POLY3"
```

9.3 Example 16 - S Plane Frequency Response of Nonrational Functions

Problem: Evaluate the open and closed loop frequency response of the following system which has a time delay. Assume that G(s) has been loaded into **SPTF$_3$**.

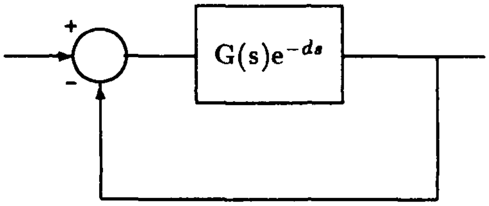


Figure 9.4: Closed Loop System with Time Delay

The following are three different methods available in LCAP2 for computing frequency responses of nonrational s plane transfer functions.

1. Use a Padé approximation (which is a rational function) for the time delay so that a rational transfer function can be computed. The desired frequency response can then be computed with the command SFREQ.
2. If an s plane transfer function consists of a rational function in series with a time delay, such as $G(S)e^{-ds}$ in this example, the command SFREQ can be used with the parameter FDELAY set equal to the time delay d.
3. The generalized s plane frequency response command, FREQS can be used to compute any function which can be coded as a complex function by the user. The name of the complex function is passed as the argument of the command FREQS.

This example will only describe the use of methods 2 and 3. Method 2 will be used to compute the open loop frequency response and method 3 will be used to compute the closed loop freouency response.

To simplify the coding of user-defined complex functions for method 3, complex function SFAUX1 in the LCAP2 source code library should be used. This function is set up so that the user can easily compute the response of any desired s plane function without the need for detail knowledge on how the interface with the frequency response routine is implemented. In this function use is made of complex function SFAUX(SPTFi) which will compute the response of transfer function SPTFi[1]. Also, the frequency parameter used for evaluating the response is available to the user as the variable U (in rad/sec). Thus, the following FORTRAN code fragment in SFAUX1 will define the closed loop transfer for this example:

```
        :
```

[1] Code in SFAUX1 is set up so that i must be between 1 and 5 if no additional code is to be written by the user. If i must be greater than 5, see comments in the SFAUX1 source code.

```
      COMPLEX CTEMP                "temporary"
      EXTERNAL SFAUX1
                    ⋮
      D= ...                       "value of delay d"
      DS=-D*U                      "U is the real frequency in rad/sec"
      IF(RAD.EQ.0)DS=DS*TWOPI      "convert DS to Hz if RAD=0"
      CTEMP=SFAUX(SPTF3)*CEXP(CMPLX(0.,DS))   "G(s)e^{-ds}, assuming G(s) is in SPTF_3"
      SFAUX1=CTEMP/(1.+CTEMP)      "closed loop transfer function"
```

The UPDATE inputs[1], which include the FORTRAN code, for this example is:

```
*IDENT idname
*INSERT START.1
      PROGRAM LCAP2(INPUT,OUTPUT,TAPE5=input,TAPE6=OUTPUT)
*CALL COMLCAP2
      EXTERNAL SFAUX1              "place here before first executable statement"
      CALL INITO                   "note that the last character is the number 0"
      CALL MINITO                  "note that the last character is the number 0"
                    ⋮
      code for loading in SPTF_3
                    ⋮
C     ENTER FREQUENCY PARAMETERS FOR AUTO MODE
      FAUTO=1
      NOMEGA=...
      OMEGA(1)=...
                    ⋮
C
C     **** COMPUTE OPEN LOOP FREQUENCY RESPONSE ****
      FDELAY=d                     "d = time delay"
      CALL SFREQ(3)                "compute frequency response of G(s)e^{-ds}"
                    ⋮
C     **** COMPUTE CLOSED LOOP FREQUENCY RESPONSE ****
C     RESET FDELAY TO ZERO SINCE TIME DELAY WILL BE COMPUTED IN FUNCTION SFAUX1
      FDELAY=0
C     DEFINE TRANSFER FUNCTION WITH FUNCTION SFAUX1
      CALL FREQS(SFAUX1)           "SFAUX1 defines the closed loop transfer function"
                    ⋮
      CALL LEXIT
      END
*DELETE SFAUX1.44                  "SFAUX1.44 is the card ident to be deleted and
```

[1] The UPDATE inputs are presented since they show the placement of the "EXTERNAL SFAUX1" statement and the UPDATE directives needed for making changes to COMPLEX FUNCTION SFAUX1.

```
                                         replaced by the following statements”
      D= ...                            “value of delay d”
      DS=-D*U                           “U is the frequency ”
      CTEMP=SFAUX(SPTF3)*CEXP(CMPLX(0.,DS))  “G(S)e^{-ds}”
C     EXPLANATION OF PREVIOUS STATEMENT:
C     SFAUX(SPTF3) IS A COMPLEX VALUE EQUAL TO THE RESPONSE AT FREQUENCY U,
C     WHERE U OF COMMON/FRQBLK/ IS THE REAL FREQUENCY BEING VARIED
C     AUTOMATICALLY BY THE LCAP2.  CEXP(CMPLX(0.,DS)) IS THE COMPLEX VALUE OF
C     THE DELAY.
C
      SFAUX1=CTEMP/(1.+CTEMP)    “closed loop transfer function”
C     SFAUX1 IS THE RETURNED VALUE OF THE COMPUTED RESPONSE
```

9.4 Example 17 IEEE CACSD Benchmark Problem - Simple Continuous Model (automated analysis method)

Solution of the simple (13 state) continuous model from the IEEE CACSD Benchmark Problems [4] is presented in this example using the LCAP2 automated analysis method based on transfer function connection blocks. The block diagram of this problem, labeled in terms of (1) LCAP2 transfer function connection blocks and (2) LCAP $\mathbf{SPTF}_i$ transfer functions, is given in Figure 9.5.

The objectives of this benchmark problem are:

- Time response:
  Apply a step input at η_{cp} and plot the output at η_p from t=0 to t=10 sec in 0.1 sec increments.
- Frequency response:
  Break the Y_4 and Y_1 feedback paths sequentially. Generate Bode plots (magnitude in dB and phase in degrees) for each break for 50 frequency points over the range $0.01 \leq \omega \leq 100$ rad/sec.

The system parameters and transfer functions are given as:

a1=.08197
a5=.2551
a8=-.4283
a9=.6645
a11=.3077
a12=3.2314
a13=.8597
a14=-.002323
a15=-.01139
a16=45.112
k1=1
k2=1.25
k3=.001244
k4=.1
k5=.01125
k6=.06
k9=29.82
g2= 1 /((1/220)*s + 1)
g3= 1 /((1/85/85)*s**2 + (2*.6/85)*s + 1)
g5= 1 /((1/170)*s + 1)
g7= 1 / s
g9=g91*g92
g91= (s + 1.5)/s
g92= ((1/150)*s + 1)/((1/50)*s + 1)
g1= 1
g4= 1

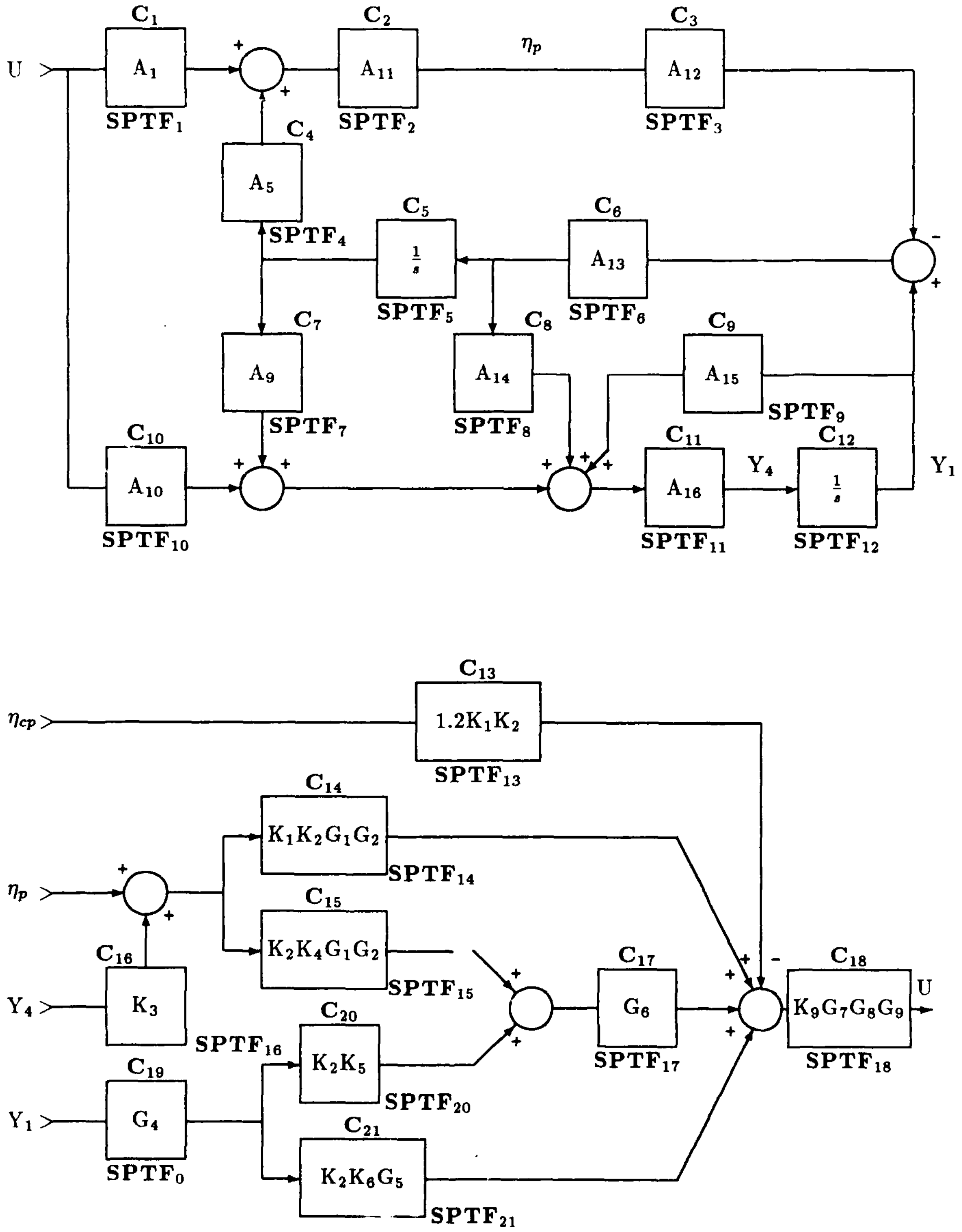


Figure 9.5: IEEE Benchmark - Block Diagram for Simple Continuous Model

The following steps were used to solve this benchmark problem:

1. Enter in constants
2. Load in s plane transfer functions
3. Define and connect transfer function blocks for closed loop configuration
4. Compute closed loop transfer function (input into $\mathbf{C}_{13}$, output = $\mathbf{C}_2$)
5. Compute closed loop time response with command ZTIME
6. Compute closed loop time response with command B1TIME
7. Change connection blocks for open loop at Y_4
8. Compute open loop transfer function with loop opened at Y_4 (input into $\mathbf{C}_{16}$, output = $\mathbf{C}_{11}$)
9. Compute frequency response with loop opened at Y_4
10. Change connection blocks for open loop at Y_1
11. Compute open loop transfer function with loop opened at Y_1 (input into $\mathbf{C}_{19}$, output = $\mathbf{C}_{12}$)
12. Compute frequency response with loop opened at Y_1

Since there are many steps involved in the solution of this problem, the PRECMP precompiler will be used to simplify the number of user input data. This data will be annotated with the step numbers shown above. If the reader has read Section 7.1 and Chapter 8, this code should be easy to follow. The PRECMP code is:

```
C     ****************************************************************
C     IEEE CACSD BENCHMARK PROBLEM - SIMPLE CONTINUOUS MODEL
C     ****************************************************************
C     1. **** ENTER IN CONSTANTS ****
C
      A1=.08197
      A2=.02296
      A3=-.004879
      A4=-.03402
      A5=.2551
      A6=.02549
      A7=-.1199
      A8=-.4283
      A9=.6645
      A10=.4078
      A11=.3077
      A12=3.2314
      A13=.8597
      A14=-.002323
      A15=-.01139
      A16=45.112
      BB1=.007429
      BB2=.07520
      BB3=.06919
      BB4=-.8272
      BB5=.03861
      BB6=-.3616
      BB7=-.3920
      BB8=-.03882
      BB9=.8064
      BB10=-.1037
      BB11=.3077
      BB12=3.2314
      BB13=.0
      BB14=.002115
      BB15=-.01139
      BB16=45.112
      BB17=.4040
      C1=-.01523
      C2=.06768
      C3=.01514
      C4=.002194
      C5=.07528
      C6=-.0002996
      C7=3030.76
      K1=1.
      K2=1.25
      K3=.001244
      K4= .1
      K5=.01125
      K6=.06
      K9=29.82
      K11=.01120
      K12=.565
      K13=2.9
      K14=.66
C
C     FOR SIMPLE MODEL
      K7=0
      K8=0
      K10=0
      G1=1.
C
C     2. **** LOAD IN S PLANE TRANSFER FUNCTIONS ****
*POLYN 0 A1
*POLYD 0 1
*SPLDC 1
```

```
C
*POLYN 0 A11
*POLYD 0 1
*SPLDC 2
C
*POLYN 0 A12
*POLYD 0 1
*SPLDC 3
C
*POLYN 0 A5
*POLYD 0 1
*SPLDC 4
C
*POLYN 0 1
*POLYD 1 0 1
*SPLDC 5
C
*POLYN 0 A13
*POLYD 0 1
*SPLDC 6
C
*POLYN 0 A9
*POLYD 0 1
*SPLDC 7
C
*POLYN 0 A14
*POLYD 0 1
*SPLDC 8
C
*POLYN 0 A15
*POLYD 0 1
*SPLDC 9
C
*POLYN 0 A8
*POLYD 0 1
*SPLDC 10
C
*POLYN 0 A16
*POLYD 0 1
*SPLDC 11
C
*POLYN 0 1.2*K1*K2
*POLYD 0 1
*SPLDC 13
C
*ROOTN K1*K2*G1
*ROOTD 1. -220.
*SPLDR 14
```

```
C
*ROOTN K2*K4*G1
*ROOTD 1. [.6,85.]
*SPLDR 15
C
*POLYN 0 K3
*POLYD 0 1
*SPLDC 16
C
*ROOTN 1. 0. -30.
*ROOTD 1. -80. [.3,160.] -1600.
*SPLDR 17
C
*ROOTN K9*1.5 -1.5 -150.
*ROOTD 1. 0. 0. -50.
*SPLDR 18
C
*POLYN 0 K2*K5
*POLYD 0 1.
*SPLDC 20
C
*ROOTN K2*K6
*ROOTD 1. -170.
*SPLDR 21
C
C     3. **** DEFINE AND CONNECT TRANSFER FUNCTION BLOCKS FOR CLOSED
C        **** LOOP CONFIGURATION
C
*B1INIT 'IEEE BENCHMARK-SIMPLE CONT. MODEL, CLOSED LOOP' ..
      'NEW' 21
C
*IYCIN 18
      INDX=1
      ISPTF=1
      NYCIN=1
*B1CEQ 'A1 BLOCK' INDX ISPTF 0 NYCIN
C
*IYCIN 1 4
*B1CEQ 'A11 BLOCK' 2 2 0 2
C
*IYCIN 2
*B1CEQ 'A12 BLOCK' 3 3 0 1
C
*IYCIN 5
*B1CEQ 'A5 BLOCK' 4 4 0 1
C
*IYCIN 6
*B1CEQ 'INTEGRATOR' 5 5 0 1
```

```
C
*IYCIN -3 12
*B1CEQ 'A13 BLOCK' 6 6 0 2
C
*IYCIN 5
*B1CEQ 'A9 BLOCK' 7 7 0 1
C
*IYCIN 6
*B1CEQ 'A14 BLOCK' 8 8 0 1
C
*IYCIN 12
*B1CEQ 'A15 BLOCK' 9 9 0 1
C
*IYCIN 18
*B1CEQ 'A8 BLOCK' 10 10 0 1
C
*IYCIN 7 8 9 10
*B1CEQ 'A16 BLOCK' 11 11 0 4
C
*IYCIN 11
*B1CEQ 'INTEGRATOR' 12 5 0 1
C
*IYCIN 0
*B1CEQ '1.2*K1*K2 BLOCK' 13 13 0 0
C
*IYCIN 2 16
*B1CEQ 'K1*K2*G1*G2 BLOCK' 14 14 0 2
C
*IYCIN 2 16
*B1CEQ 'K2*K4*G1*G3 BLOCK' 15 15 0 2
C
*IYCIN 11
*B1CEQ 'K3 BLOCK' 16 16 0 1
C
*IYCIN 15 20
*B1CEQ 'G6 BLOCK' 17 17 0 2
C
*IYCIN -13 14 17 21
*B1CEQ 'K9*G7*G8*G9 BLOCK' 18 18 0 4
C
*IYCIN 12
*B1CEQ 'G4 BLOCK' 19 0 0 1
C
*IYCIN 19
*B1CEQ 'K2*K5 BLOCK' 20 20 0 1
C
*IYCIN 19
*B1CEQ 'K2*K6*G5 BLOCK' 21 21 0 1
```

```
C
*B1END
C
C     **** CONNECTION OF TRANSFER FUNCTION BLOCKS COMPLETED ****
C
C     4. **** COMPUTE CLOSED LOOP TRANSFER FUNCTION ****
C
      UCIN=13
      UMAGN=1.
      YCOUT=2
      PRINT*,'CLOSED LOOP TRANSFER FUNCTION WILL BE IN SPTF30'
C     SET PRNMTRX=1 FOR PRINTOUT OF COMPUTED STATE SPACE MATRICES
      PRNMTRX=1
*B1TF 30
C
C     5. **** COMPUTE CLOSED LOOP TIME RESPONSE WITH COMMAND STIME ****
C
      TMAGN=1.
      TEND=10.
      TDELT=.01
      HRDCPY=1
      TITLE1='EXAMPLE 17, IEEE BENCHMARK, SIMPLE CONTINUOUS MODEL'
      TITLE2='TIME RESPONSE OF ETA-P/ETA-RHO'
*STIME 30
C
C     6. **** COMPUTE CLOSED LOOP TIME RESPONSE WITH COMMAND B1TIME ****
      UCIN=13
      UMAGN=1.
      YCOUT=2
      CALL B1TIME()
C
C     7. **** CHANGE CONNECTION DATA FOR OPEN LOOP AT Y4 ****
C
*B1INIT 'IEEE BENCHMARK-SIMPLE CONT. MODEL, OPEN LOOP AT Y4' ..
          'OLD' 21
*B1CEQ 'K3 BLOCK' 16 16 0 0  ! OPEN LOOP AT Y4
*B1END
C
C     8. **** COMPUTE OPEN LOOP T.F. WITH LOOP OPENED AT Y4 ****
      UCIN=16
      UMAGN=1.
      YCOUT=11
      PRINT*,'OPEN LOOP T.F. AT Y4 WILL BE STORED IN SPTF31'
*B1TF 31
C
C     9. **** COMPUTE FREQUENCY RESPONSE WITH LOOP OPENED AT Y4 ***
      NOMEGA=4
*OMEGA .1 1. 10. 100.
```

```
      FNICO=1
      TITLE1='OPEN LOOP TRANSFER FUNCTION AT Y4'
      CALL SFREQ(31)
C
C     10. **** CHANGE CONNECTION DATA FOR OPEN LOOP AT Y1 ****
*B1INIT 'IEEE BENCHMARK-SIMPLE CONT. MODEL, OPEN LOOP AT Y1' ..
          'OLD' 21
*IYCIN 11
*B1CEQ 'K3 BLOCK' 16 16 1  ! CLOSE LOOP AT Y4
C
*B1CEQ 'G4 BLOCK' 19 0 0  ! OPEN LOOP AT Y1
*B1END
C
C     11. **** COMPUTE OPEN LOOP T.F. WITH LOOP OPENED AT Y1 ****
      UCIN=19
      UMAGN=1.
      YCOUT=12
      PRINT*,'OPEN LOOP T.F. AT Y1 WILL BE STORED IN SPTF32'
*B1TF 32
C
C     12. **** COMPUTE FREQUENCY RESPONSE WITH LOOP OPENED AT Y1 ****
      TITLE1='OPEN LOOP TRANSFER FUNCTION AT Y1'
      CALL SFREQ(32)
```

Since the complete output of the example is too long to be included in this report, only selected output is shown. The output from Step 4 for the command B1TF is:

```
CLOSED LOOP TRANSFER FUNCTION WILL BE IN SPTF30

IEEE BENCHMARK-SIMPLE CONT. MODE1, CLOSED LOOP
NUMBER OF CONTINUOUS BLOCKS = 21
BLOCK    DESCRIPTION

C1       A1 BLOCK
C2       A11 BLOCK
C3       A12 BLOCK
C4       A5 BLOCK
C5       INTEGRATOR
C6       A13 BLOCK
C7       A9 BLOCK
C8       A14 BLOCK
C9       A15 BLOCK
C10      A8 BLOCK
C11      A16 BLOCK
C12      INTEGRATOR
```

```
C13      1.2*K1*K2 BLOCK
C14      K1*K2*G1*G2 BLOCK
C15      K2*K4*G1*G3 BLOCK
C16      K3 BLOCK
C17      G6 BLOCK
C18      K9*G7*G8*G9 BLOCK
C19      G4 BLOCK
C20      K2*K5 BLOCK
C21      K2*K6*G5 BLOCK

BLOCK     TRANSFER FUNCTION          INPUTS FROM BLOCKS

C1           SPTF1              C18
C2           SPTF2              C1   , C4
C3           SPTF3              C2
C4           SPTF4              C5
C5           SPTF5              C6
C6           SPTF6             -C3   , C12
C7           SPTF7              C5
C8           SPTF8              C6
C9           SPTF9              C12
C10          SPTF10             C18
C11          SPTF11             C7   , C8   , C9   , C10
C12          SPTF5              C11
C13          SPTF13             HAS NO INPUT
C14          SPTF14             C2   , C16
C15          SPTF15             C2   , C16
C16          SPTF16             C11
C17          SPTF17             C15 , C20
C18          SPTF18            -C13 , C14 , C17 , C21
C19          SPTF0              C12
C20          SPTF21             C19
C21          SPTF22             C19

----------------------------------

INPUT U IS CONNECTED TO BLOCK C13
MAGNITUDE OF U =  .10000E+01
BLOCK C2   IS THE OUTPUT

*********************************************************************
*    B1TF - BLOCK1: FIND TRANSFER FUNCTION FROM C13  TO C2          *
*                     AND STORE IN SPTF30                           *
*********************************************************************

COMPUTED CONTINUOUS DYNAMICS STATE SPACE REPRESENTATION IS

     D(X)/DT  = A * X  +  B * U
```

```
          Y     = C * X  +  D * U

WHERE X IS A SET OF S-PLANE STATE VARIABLES OF LENGTH 13

      A IS A (13,13) MATRIX
      B IS A (13, 1) MATRIX
      C IS A (21,13) MATRIX
      D IS A (21, 1) MATRIX
      Y IS THE OUTPUT VECTOR OF LENGTH 21

(THE I-TH ELEMENT OF Y IS THE OUTPUT OF THE I-TH CONNECTION BLOCK)
MATRIX A         IS
( 1, 1)-.21805980E+00 ( 1, 2) .85970000E+00 ( 1,10)-.70068058E-01
( 2, 1) .29999776E+02 ( 2, 2)-.60391809E+00 ( 2,10)-.19314127E+02
( 3, 1) .31848847E+02 ( 3, 2)-.20660038E+00 ( 3, 3)-.22000000E+03
( 3,10) .32873369E+00 ( 4, 4)-.10200000E+03 ( 4, 5) .10000000E+01
( 5, 1) .10459451E+03 ( 5, 2)-.67849443E+00 ( 5, 4)-.72250000E+04
( 5,10) .10795913E+01 ( 6, 6)-.17760000E+04 ( 6, 7) .10000000E+01
( 7, 2) .15360000E+07 ( 7, 4) .10922667E+09 ( 7, 6)-.31488000E+06
( 7, 8) .10000000E+01 ( 8, 2) .46080000E+08 ( 8, 4) .32768000E+10
( 8, 6)-.55296000E+08 ( 8, 9) .10000000E+01 ( 9, 6)-.32768000E+10
(10, 3) .99400000E+01 (10, 6) .99400000E+01 (10,10)-.50000000E+02
(10,11) .10000000E+01 (10,13) .99400000E+01 (11, 3) .15059100E+04
(11, 6) .15059100E+04 (11,12) .10000000E+01 (11,13) .15059100E+04
(12, 3) .22365000E+04 (12, 6) .22365000E+04 (12,13) .22365000E+04
(13, 2) .12750000E+02 (13,13)-.17000000E+03
MATRIX B         IS
(10, 1)-.14910000E+02 (11, 1)-.22588650E+04 (12, 1)-.33547500E+04

MATRIX C         IS
( 1,10) .81970000E-01 ( 2, 1) .78494270E-01 ( 2,10) .25222169E-01
( 3, 1) .25364638E+00 ( 3,10) .81502917E-01 ( 4, 1) .25510000E+00
( 5, 1) .10000000E+01 ( 6, 1)-.21805980E+00 ( 6, 2) .85970000E+00
( 6,10)-.70068058E-01 ( 7, 1) .66450000E+00 ( 8, 1) .50655291E-03
( 8, 2)-.19970831E-02 ( 8,10) .16276810E-03 ( 9, 2)-.11390000E-01
(10,10)-.42830000E+00 (11, 1) .29999776E+02 (11, 2)-.60391809E+00
(11,10)-.19314127E+02 (12, 2) .10000000E+01 (14, 3) .10000000E+01
(15, 4) .10000000E+01 (16, 1) .37319721E-01 (16, 2)-.75127411E-03
(16,10)-.24026774E-01 (17, 6) .10000000E+01 (18,10) .10000000E+01
(19, 2) .10000000E+01 (20, 2) .14062500E-01 (21,13) .10000000E+01
MATRIX D         IS
(13, 1) .15000000E+01

THE COMMON ROOTS ELIMINATED ARE
NO.         REAL             IMAG.              OMEGA            ZETA

 1      -1600.0000        0.
 2      -220.00000        0.
```

```
   3     -170.00000       0.

                              THE NUMERATOR ROOTS OF SROOT30  ARE
 NO.          REAL              IMAG.               OMEGA              ZETA

   1     -48.000000         152.63027          160.00000          .30000000
   2     -48.000000        -152.63027          160.00000          .30000000
   3     -51.000000         68.000000          85.000000          .60000000
   4     -51.000000        -68.000000          85.000000          .60000000
   5     -150.00000       0.
   6     -80.000000       0.
   7      8.5046654       0.
   8     -9.1085835       0.
   9     -1.5000000       0.

                              LOW ORDER NONZERO COEFFICIENT =     .582037682E+22

                              THE DENOMINATOR ROOTS OF SROOT30  ARE

 NO.          REAL              IMAG.               OMEGA              ZETA

   1     -44.092603         150.02933          156.37441          .28196815
   2     -44.092603        -150.02933          156.37441          .28196815
   3     -51.266369         67.429071          84.704901          .60523498
   4     -51.266369        -67.429071          84.704901          .60523498
   5     -65.215104         13.666998          66.631799           97873846
   6     -65.215104        -13.666998          66.631799          .97873846
   7     -.52434498         4.7138755          4.7429485          .11055254
   8     -.52434498        -4.7138755          4.7429485          .11055254
   9     -5.7150865       0.
  10     -.96347515       0.

                              LOW ORDER NONZERO COEFFICIENT =     .57741883E+22

DEGREE OF NUMERATOR OF SPTF30  IS  9    (COEFFICIENTS IN ASCENDING ORDER)
5.803768209021E+21 4.038871396247E+21 4.034049515012E+19 -5.128910870111E+19
-1.891779440678E+18 -3.333511114137E+16 -3.524184416543E+14  -2.272432391348E+12
-9.678878953207E+9 -22503582.38103

DEGREE OF DENOMINATOR OF SPTF30  IS 10  (COEFFICIENTS IN ASCENDING ORDER)
5.774188337049E+21 7.545575848071E+21 1.98128939038E+21 4.445690030619E+20
6.533523088991E+19 2.577851657259E+18 5.026005806442E+16 5.026005806442E+16
5.890697382345E+14 4.338767884133E+12 1.968115955848E+10 59843817.12135

BODE GAIN =    1.0051228
```

The above output for the command B1TF summarizes all of the connection blocks so that the analyst can easily verify that the system is correctly modeled. The computed state space matrices are printed out sinre the parameter PRNMTRX was set to 1. Thirteen states were computed from the dynamics associated with connection blocks C_5, C_{12}, C_{14}, C_{18}, and C_{21} (they correspond to states X(1), X(1), X(3), X(4), X(6), X(10), and X(13), respectively). All other connection blocks are algebraic blocks which do not affect the size of the system matrix. The output of each C_i block is computed using matrices C and D. For example, the output connection block C_2 is the sum of C(2,1)*X(1) and C(2,10)*X(10) which are the output of connection blocks C_5 and C_{10}, respectively.

The computed closed loop transfer function is stored in **SPTF**$_{30}$. This transfer function is used in Step 5 to compute the inverse Laplace transform and time response. Portions of the output from this step is given by:

```
************************************************************
*     STIME = TIME RESPONSE OF S-PLANE TRANSFER            *
*             FUNCTION 30                                  *
************************************************************

COMPUTE STEP RESPONSE
SCALE INPUT BY TMAGN, (TMAGN =   1.00000     )

 NO.              ROOT                     PARTIAL FRACTION COEFFICIENT

 1      -44.092603         150.02933       -.12032418E-04   -.90545444E-04
 2      -44.092603        -150.02933       -.12032418E-04    .90545444E-04
 3      -51.266369         67.429071       -.34822493E-04    .59647045E-04
 4      -51.266369        -67.429071       -.34822493E-04   -.59647045E-04
 5      -65.215104         13.666998       -.58203173E-02    .14050461E-02
 6      -65.215104        -13.666998       -.58203173E-02   -.14050461E-02
 7      -.52434498         4.7138755       -.19165032        .29006868
 8      -.52434498        -4.7138755       -.19165032       -.29006868
 9      -5.7150865        0.               -.17370408       0.
10      -.96347515        0.               -.43638372       0.
11      0.                0.                1.0051228       0.

ANALYTICAL SOLUTION IS THE SUMMATION OF THE FOLLOWING  7 TERMS


                                            (  1.0051     ) * T** 0
((-.2406E-04)*(COS( 150.    *T))+( .1811E-03)*(SIN( 150.    *T)))*E**(-44.09*T)
((-.6964E-04)*(COS( 67.4    *T))+(-.1193E-03)*(SIN( 67.4    *T)))*E**(-51.27*T)
((-.1164E-01)*(COS( 13.7    *T))+(-.2810E-02)*(SIN( 13.7    *T)))*E**(-65.22*T)
((-.3833    )*(COS( 4.71    *T))+(-.5801    )*(SIN( 4.71    *T)))*E**(-.5243*T)
                                      ( -.17370    ) * E**( -5.7151  * T)
                                      ( -.43638    ) * E**( -.96348  * T)
```

Neither the tabular output of the response nor the low resolution printer plot is presented; however, the high resolution plot is presented in Appendix H.

The step response computed by state space method in Step 6 is identical to the results of Step 5. The computed state space matrices are the same as the ones computed in Step 4.

After the loop at Y_4 is opened in Step 7, the command B1TF is used to compute the the open loop transfer function. Since the format of the output is similar to that presented in Step 4, only the final results of this open loop transfer function are presented below:

```
INPUT U IS CONNECTED TO BLOCK C16
MAGNITUDE OF U =  .10000E+01
BLOCK C11  IS THE OUTPUT

**********************************************************************
*    B1TF - BLOCK1: FIND TRANSFER FUNCTION FROM C16  TO C11          *
*                   AND STORE IN SPTF31                              *
**********************************************************************

COMPUTED CONTINUOUS DYNAMICS STATE SPACE REPRESENTATION IS

     D(X)/DT  = A * X  +  B * U
        Y     = C * X  +  D * U

WHERE X IS A SET OF S-PLANE STATE VARIABLES OF LENGTH 13

      A IS A (13,13) MATRIX
      B IS A (13, 1) MATRIX
      C IS A (21,13) MATRIX
      D IS A (21, 1) MATRIX
      Y IS THE OUTPUT VECTOR OF LENGTH 21

(THE I-TH ELEMENT OF Y IS THE OUTPUT OF THE I-TH CONNECTION BLOCK)
MATRIX A          IS
( 1, 1)-.21805980E+00 ( 1, 2) .85970000E+00 ( 1,10)-.70068058E-01
( 2, 1) .29999776E+02 ( 2, 2)-.60391809E+00 ( 2,10)-.19314127E+02
( 3, 1) .21585924E+02 ( 3, 3)-.22000000E+03 ( 3,10) .69360965E+01
( 4, 4)-.10200000E+03 ( 4, 5) .10000000E+01 ( 5, 1) .70890138E+02
( 5, 4)-.72250000E+04 ( 5,10) .22778771E+02 ( 6, 6)-.17760000E+04
( 6, 7) .10000000E+01 ( 7, 2) .15360000E+07 ( 7, 4) .10922667E+09
( 7, 6)-.31488000E+06 ( 7, 8) .10000000E+01 ( 8, 2) .46080000E+08
( 8, 4) .32768000E+10 ( 8, 6)-.55296000E+08 ( 8, 9) .10000000E+01
( 9, 6)-.32768000E+10 (10, 3) .99400000E+01 (10, 6) .99400000E+01
(10,10)-.50000000E+02 (10,11) .10000000E+01 (10,13) .99400000E+01
(11, 3) .15059100E+04 (11, 6) .15059100E+04 (11,12) .10000000E+01
(11,13) .15059100E+04 (12, 3) .22365000E+04 (12, 6) .22365000E+04
```

```
(12,13) .22365000E+04 (13, 2) .12750000E+02 (13,13)-.17000000E+03
MATRIX B          IS
( 3, 1) .34210000E+00 ( 5, 1) .11234875E+01
MATRIX C          IS
( 1,10) .81970000E-01 ( 2, 1) .78494270E-01 ( 2,10) .25222169E-01
( 3, 1) .25364638E+00 ( 3,10) .81502917E-01 ( 4, 1) .25510000E+00
( 5, 1) .10000000E+01 ( 6, 1)-.21805980E+00 ( 6, 2) .85970000E+00
( 6,10)-.70068058E-01 ( 7, 1) .66450000E+00 ( 8, 1) .50655291E-03
( 8, 2)-.19970831E-02 ( 8,10) .16276810E-03 ( 9, 2)-.11390000E-01
(10,10)-.42830000E+00 (11, 1) .29999776E+02 (11, 2)-.60391809E+00
(11,10)-.19314127E+02 (12, 2) .10000000E+01 (14, 3) .10000000E+01
(15, 4) .10000000E+01 (17, 6) .10000000E+01 (18,10) .10000000E+01
19, 2) .10000000E+01 (20, 2) .14062500E-01 (21,13) .10000000E+01
MATRIX D          IS
16, 1) .12440000E-02

THE COMMON ROOTS ELIMINATED ARE

NO.         REAL             IMAG.             OMEGA            ZETA

 1     -170.00000       0.
                    THE NUMERATOR ROOTS OF SROOT31  ARE

NO.         REAL             IMAG.             OMEGA            ZETA

 1      40.155541        493.26742        494.89919       -.81138829E-01
 2      40.155541       -493.26742        494.89919       -.81138829E-01
 3     -1721.1005       0.
 4     -202.28422       0.
 5     -150.00000       0.
 6     -22.689956       0.
 7     -12.236364       0.
 8     -.32689340       0.
 9     -1.5000000       0.
10     0.               0.

                    LOW ORDER NONZERO COEFFICIENT =   -.19441954E+20

                    THE DENOMINATOR ROOTS OF SROOT31  ARE

NO.         REAL             IMAG.             OMEGA            ZETA

 1     -43.556200        153.17063        159.24316         .27352006
 2     -43.556200       -153.17063        159.24316         .27352006
 3     -52.948241        51.908661        74.148671         .71408214
 4     -52.948241       -51.908661        74.148671         .71408214
 5     -64.236758        32.236076        71.871592         .89377119
 6     -64.236758       -32.236076        71.871592         .89377119
 7      -.41681705        5.0261118        5.0433655        .82646606E-01
 8      -.41681705       -5.0261118        5.0433655        .82646606E-01
 9      -1599.9227        0.
10      -220.13034        0.
11      -5.4775769        0.
12      -.96106888        0.

                    LOW ORDER NONZERO COEFFICIENT =     .57741883E+22

DEGREE OF NUMERATOR OF SPTF31  IS 10    (COEFFICIENTS IN ASCENDING ORDER)
0. -1.944195413229E+19 -7.511257462116E+19 -4.9721269018E+19 -5.830992556724E+18
-2.070234735638E+17 -1.905771067787E+15 -5.9831568368E+12 -8.80551195449E+9
-22663264.14444 -11165.12162595

DEGREE OF DENOMINATOR OF SPTF31  IS 12  (COEFFICIENTS IN ASCENDING ORDER)
5.774188337049E+21 7.555988560843E+21 1.945206599055E+21 4.051131250508E+20
6.180842727466E+19 2.709895512996E+18 6.186825255004E+16 8.50272624298E+14
7.518455107493E+12 4.376503808893E+10 173917735.4379 365327.90872 170.0142370976

BODE GAIN =    -.33070454E-02
```

In Step 9 the frequency response with the loop opened at Y_4 is computed.[1] The printer output for this case is not shown. However, the high resolution plot is presented in Appendix H. Note that the response as computed is the negative of open loop frequency response normally used to assess stability.

Similarly, the open loop loop transfer function with the loop opened at Y_1 (Step 11) is presented below:

```
INPUT U IS CONNECTED TO BLOCK C20
MAGNITUDE OF U =  .10000E+01
BLOCK C12  IS THE OUTPUT

***********************************************************************
*    B1TF - BLOCK1: FIND TRANSFER FUNCTION FROM C20  TO C12           *
*                       AND STORE IN SPTF32                           *
***********************************************************************

COMPUTED CONTINUOUS DYNAMICS STATE SPACE REPRESENTATION IS

     D(X)/DT  = A * X  +  B * U
        Y     = C * X  +  D * U
```

[1] The frequency response could also have been computed with command B1FREQ which would have yielded identical results.

```
WHERE X IS A SET OF S-PLANE STATE VARIABLES OF LENGTH 13

      A IS A (13,13) MATRIX
      B IS A (13, 1) MATRIX
      C IS A (21,13) MATRIX
      D IS A (21, 1) MATRIX
      Y IS THE OUTPUT VECTOR OF LENGTH 21

(THE I-TH ELEMENT OF Y IS THE OUTPUT OF THE I-TH CONNECTION BLOCK)
MATRIX A          IS
( 1, 1)-.21805980E+00 ( 1, 2) .85970000E+00 ( 1,10)-.70068058E-01
( 2, 1) .29999776E+02 ( 2, 2)-.60391809E+00 ( 2,10)-.19314127E+02
( 3, 1) .31848847E+02 ( 3, 2)-.20660038E+00 ( 3, 3)-.22000000E+03
( 3,10) .32873369E+00 ( 4, 4)-.10200000E+03 ( 4, 5) .10000000E+01
( 5, 1) .10459451E+03 ( 5, 2)-.67849443E+00 ( 5, 4)-.72250000E+04
( 5,10) .10795913E+01 ( 6, 6)-.17760000E+04 ( 6, 7) .10000000E+01
( 7, 4) .10922667E+09 ( 7, 6)-.31488000E+06 ( 7, 8) .10000000E+01
( 8, 4) .32768000E+10 ( 8, 6)-.55296000E+08 ( 8, 9) .10000000E+01
( 9, 6)-.32768000E+10 (10, 3) .99400000E+01 (10, 6) .99400000E+01
(10,10)-.50000000E+02 (10,11) .10000000E+01 (10,13) .99400000E+01
(11, 3) .15059100E+04 (11, 6) .15059100E+04 (11,12) .10000000E+01
(11,13) .15059100E+04 (12, 3) .22365000E+04 (12, 6) .22365000E+04
(12,13) ~2365000E+04 (13,13)-.17000000E+03
MATRIX B          IS
( 7, 1) .15360000E+07 ( 8, 1) .46080000E+08 (13, 1) .12750000E+02
MATRIX C          IS
( 1,10) .81970000E-01 ( 2, 1) .78494270E-01 ( 2,10) .25222169E-01
( 3, 1) .25364638E+00 ( 3,10) .81502917E-01 ( 4, 1) .25510000E+00
( 5, 1) .10000000E+01 ( 6, 1)-.21805980E+00 ( 6, 2)  85970000E+00
( 6,10)-.70068058E-01 ( 7, 1) .66450000E+00 ( 8, 1) .50655291E-03
( 8, 2)-.19970831E-02 ( 8,10) .16276810E-03 ( 9, 2)-.11390000E-01
(10,10)-.42830000E+00 (11, 1) .29999776E+02 (11, 2)-.60391809E+00
(11,10)-.19314127E+02 (12, 2) .10000000E+01 (14, 3) .10000000E+01
(15, 4) .10000000E+01 (16, 1) .37319721E-01 (16, 2)-.75127411E-03
(16,10)-.24026774E-01 (17, 6) .10000000E+01 (18,10) .10000000E+01
(21,13) .10000000E+01
MATRIX D          IS
(19, 1) .10000000E+01 (20, 1) .14062500E-01

THE COMMON ROOTS ELIMINATED ARE

NO.        REAL             IMAG.              OMEGA            ZETA

 1      -220.00000        0.

                THE NUMERATOR ROOTS OF SROOT32  ARE

NO.        REAL             IMAG.              OMEGA            ZETA
```

```
  1     -51.000000        68.000000        85.000000        .60000000
  2     -51.000000       -68.000000        85.000000        .60000000
  3     -122046.64       0.
  4     -168.29727       0.
  5     -150.00000       0.
  6     -25.365837       0.
  7     -6.2892356       0.
  8     -.32689340       0.
  9     -1.5000000       0.

                    LOW ORDER NONZERO COEFFICIENT =   -.93771483E+21

                    THE DENOMINATOR ROOTS OF SROOT32  ARE

NO.        REAL            IMAG.             OMEGA            ZETA

  1     -47.940879        152.78117       160.12624        .29939427
  2     -47.940879       -152.78117       160.12624        .29939427
  3     -51.288702        67.552615        84.816783       .60469992
  4     -51.288702       -67.552615        84.816783       .60469992
  5      3.1461972        2.7385314        4.1711043      -.75428399
  6      3.1461972       -2.7385314        4.1711043      -.75428399
  7     -1599.9998       0.
  8     -79.544021       0.
  9     -50.114929       0.
 10     -5.9243239       0.
 11     -1.0664041       0.
 12     -170.00000       0.

                    LOW ORDER NONZERO COEFFICIENT =    .48364735E+22

DEGREE OF NUMERATOR OF SPTF32  IS  9    (COEFFICIENTS IN ASCENDING ORDER)
-9.377148295316E+20 -3.704843035743E+21 -2.710203577929E+21 -4.724926275247E+20
-2.365644402255E+19 -4.695048625545E+17 -5.210858267483E+15 -2.98665864912E+13
-6.596778299719E+10 -538510.6893547

DEGREE OF DENOMINATOR OF SPTF32  IS 12  (COEFFICIENTS IN ASCENDING ORDER)
4.836473507518E+21 3.878304710069E+21 -6.797951649868E+20 -1.500417076754E+19

4.457877792923E+19 2.535107790608E+18 6.206314705428E+16 8.957164415011E+14
8.29058307049E+12 5.007812145866E+10 203857706.4704 461751.5906631 220.0057246298

BODE GAIN =   -.19388400
```

In Step 12 the frequency response with the loop opened at Y_1 is computed. The printer output for this case is not shown. However, the high resolution plot is presented in Appendix H. Note that the response as computed is the negative of the open loop frequency response normally used to assess stability. If the frequency response of -$\mathbf{SPTF}_{32}$ were computed instead, the 180 degree crossover frequency at 4.7 rad/sec would yield a gain margin of 1.37 db.

The results of this benchmark problem are essentially identical to the results given in Ref. 4. To compare the system eigenvalues with those of Ref. 4 the system eigenvalues could have been obtained with the command B1EIG. However, since the closed loop transfer function was computed in Step 4, the eigenvalues are equal to the denominator roots of $\mathbf{SPTF}_{30}$ plus the three common roots at -1600., -220. and -170. which were eliminated.

9.5 Example 18 IEEE CACSD Benchmark Problem No. 3, 2-Rate Model (automated analysis method)

Solution of the 2-rate sampled-data model from the IEEE CACSD Benchmark Problem No. 3 [5] is presented in this example using the LCAP2 automated analysis method based on transfer function connection blocks. This automated method will show the ease in setting up the analysis to solve for the objectives of the benchmark problem. The block diagram of this problem, labeled in terms of (1) LCAP2 transfer function connection blocks and (2) LCAP2 **SPTF**$_i$ and **ZPTF**$_i$ transfer functions, is given in Figure 9.6. This figure includes a unity discrete connection block $\mathbf{D_7}$ used to define the sampled output of G_1 at the slower sampling rate and a unity discrete connection block D_6 to be used in defining an open loop transfer function.

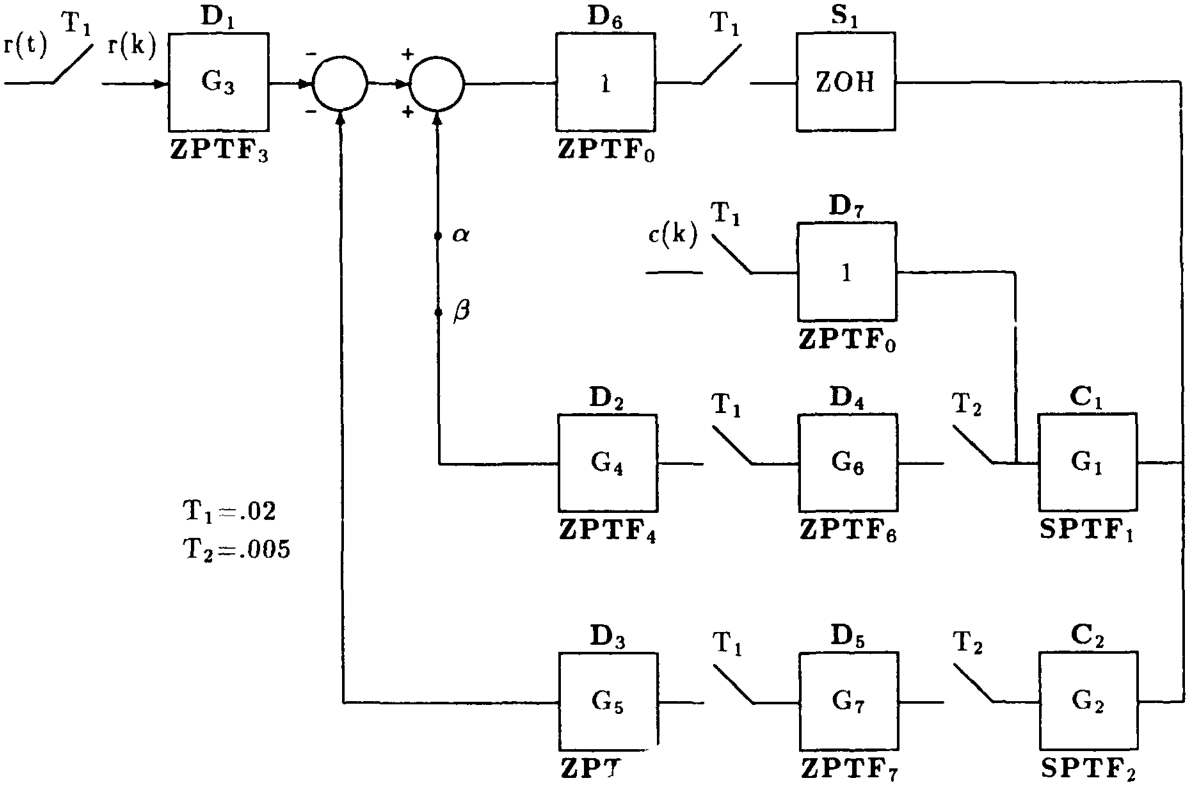


Figure 9.6: Block Diagram for IEEE Benchmark 2-Rate Model

Conventions defined by the benchmark problem

1. All samplers are ideal. This means that the samplers reach their values instantaneously, and all of the fast samplers are synchronized with the slow samplers.
2. It is important that the simulation protocol be defined so that there is a unique answer. The main difficulty is to insure that no extra delays are introduced. Assume that there is no computational delay. A subsystem with the same number of poles and zeros, as are most in this problem, has a direct connection between the input and output. When two or more samplers coincide, this direct path must propagate through the samplers.

3. The zero-order hold (ZOH) block updates at the slowest sampling rate and provides a constant input to the continuous system until the next slow sampling time occurs.

4. The continuous transfer function $G_1(s)$ and $G_2(s)$ have the same denominator and should be considered as representing a single second-order system having one input and two outputs. Do not introduce extra states by treating them as separate blocks[1].

Problem statement:

Step A: Generate the time response to a continuous-time unit step input r(t), with zero initial conditions throughout. Plot and tabulate the continuous-time output of c(t) for the interval $0 \leq t \leq 4.0$ seconds. The plot should use time increments of at most 0.02 sec. The values of c(t) should be tabulated for every 0.5 sec with at least 5 digits.

Step B:

1. Determine an equivalent discrete-time model in state space or transfer function form for the closed loop system based on the basic sampling interval of 0.02 sec. The input is to be the sample values r(k) and the output is c(k), the sample values of the continuous output c(t). If a state space model is used, identify the states in terms of the states of the transfer functions of the individual blocks.

2. Give the eigenvalues of the A matrix of this model (at least 5 digits). List them in order of increasing magnitude.

3. Find the poles and zeros, after any cancellations have been removed, of the transfer function C(z)/R(z).

Step c: Construct a model where the inner feedback loop is broken between the points α and β. The input is the discrete-time signal at α and the output is the signal at β, both of which are sampled at the basic time interval of 0.02 sec. The input reference input r(k) is identically zero. The outer feedback loop remains in place.

1. Determine the poles and zeros of the discrete transfer function from the input at α to the output at β, denoted as H(z), after any cancellations have taken place.

2. Plot and tabulate the frequency response that is found by evaluating H(z) around the upper half of the unit circle. Use dB ($20 \log_{10}$) for magnitudes, degrees for phase, and Hertz for frequency. The plots should be in Bode form, with 200 frequency points running from 0.01 Hz to 25 Hz. (At the basic sampling interval of 0.02 sec., the point z=-1 corresponds to 50π rad/sec or 25 Hz). Include numerical values (about 4 digits) of magnitude and phase for the frequencies: 0.01, 0.1, 1.0. and 25 Hz.

The system parameters and transfer functions are given as:

T_1=1/50; T_2=1/200
***continuous transfer functions
G_1 = -15 (s + 1.5) / (s**2 + 6*s + 13)

[1] In the solution of this problem in Examples 18 and 19, this was not adhered to since it is easier to model the problem as two different transfer functions with the same denominator. Extra states generated will not present any computational difficulties for this problem; it will though, generate common roots in transfer functions which will be automatically eliminated by the program.

G_2 = (- .1*s + 9) / (s**2 + 6*s + 13)
***discrete transfer functions
G_3 = .2625 (z - .9048) / (z -1)
G_4 = 1.7 (z - .9608) / (z - .6667)
G_5 = 1.035 (z - .9324) / (z - 1)
G_6 = .1564 (z + 1) / (z - .6873)
G_7 = .0362 (z + 1) / (z - .9277)

The first part of this benchmark problem is to compute the time response at the output of the continuous block $G_1(s)$. Unity discrete connection block $\mathbf{C}_7$ connected to the output of the continuous block is used as a fictitious or mathematical sampler to define the sampled output at the slower sampling rate. Thus, the time response from block $\mathbf{D}_1$ to $\mathbf{D}_7$ is to be computed. Two different methods will be used to compute this response. The first uses command ZTIME and the second uses command B2TIME.

Solution to the first part of this benchmark problem will solve the second part of the problem since both a state space and a transfer function form for the closed loop transfer function will be computed along with the eigenvalues, poles, and zeros.

The last part of this benchmark problem is to compute the open loop[1] frequency response of the system by breaking the loop between points α and β.

The following steps were used to solve this benchmark problem:

1. Load in s plane transfer functions
2. Load in z plane transfer functions
3. Enter in sampling periods
4. Define and connect transfer function blocks for closed loop configuration
5. Compute closed loop transfer function
6. Compute closed loop time response using command ZTIME
7. Compute closed loop time response using command B2TIME
8. Change connection blocks for open loop configuration
9. Compute open loop transfer function
10. Compute open loop frequency response

Since there are many steps involved in the solution of this problem, the PRECMP precompiler will be used. The PRECMP code is:

[1] The open loop transfer function as defined by Ref. 5 is not the usual definition of an open loop transfer function used for stability analysis. To properly read out gain and phase margins at the crossover frequencies, the negative of the open loop transfer function defined by this benchmark problem would be used instead.

```
C     IEEE CACSD BENCHMARK PROBLEM NO. 3, 2-RATE MODEL
C
C     1.  **** LOAD IN S PLANE TRANSFER FUNCTIONS ****
*POLYN 1 -15*1.5 -15
*POLYD 2 13 6 1
*SPLDC 1          !  LOAD IN G1(S)
C
*POLYN 1 9 -.1
*POLYD 2 13 6 1
*SPLDC 2          !  LOAD IN G2(S)
C
C     2.  **** LOAD IN Z PLANE TRANSFER FUNCTIONS ****
*POLYN 1 -.2625*.9048 .2625
*POLYD 1 -1 1
*ZPLDC 3          !  LOAD IN G3(Z)
C
*POLYN 1 -1.7*.9608 1.7
*POLYD 1 -.6667 1
*ZPLDC 4          !  LOAD IN G4(Z)
C
*POLYN 1 -1.035*.9324 1.035
*POLYD 1 -1 1
*ZPLDC 5          !  LOAD IN G5(Z)
C
*POLYN 1 .1564 .1564
*POLYD 1 -.6873 1
*ZPLDC 6          !  LOAD IN G6(Z)
C
*POLYN 1 .0362 .0362
*POLYD 1 -.9277 1
*ZPLDC 7          !  LOAD IN G7(Z)
C
C     3.  **** ENTER SAMPLING PERIODS ****
      T1=1./50.
      T2=1./200.
C
C     4.  **** DEFINE AND CONNECT TRANSFER FUNCTION BLOCKS FOR CLOSED
C         **** LOOP CONFIGURATION
      DELAY=0
C     NOTE: DELAY OPTION NOT IMPLEMENTED YET
C
*B2INIT 'IEEE BENCHMARK PROBLEM NO. 3, 2-RATE, CLOSED LOOP' ..
      'NEW' 2 7 1
C
*IXSIN 1
      INDX=1
      ISPTF=1
      NYCIN=0
      NXSIN=1
*B2CEQ 'G1 BLOCK' INDX ISPTF DELAY 0 NYCIN NXSIN
C
*IXSIN 1
*B2CEQ 'G2 BLOCK' 2 2 DELAY 0 0 1
C
*B2DEQ 'G3 BLOCK' 1 3 T1 DELAY 0 0 0 0
C
*IYDIN 4
*B2DEQ 'G4 BLOCK' 2 4 T1 DELAY 0 0 1 0
C
*IYDIN 5
*B2DEQ 'G5 BLOCK' 3 5 T1 DELAY 0 0 1 0
C
*IYCIN 1
*B2DEQ 'G6 BLOCK' 4 6 T2 DELAY 0 1 0 0
C
*IYCIN 2
*B2DEQ 'G7 BLOCK' 5 7 T2 DELAY 0 1 0 0
C
*IYDIN -1 2 -3
*B2DEQ 'SUMMER BLOCK' 6 0 T1 DELAY 0 1 0 0
C
*IYCIN 1
*B2DEQ 'T1 SAMPLER FOR G1(S) BLOCK' 7 0 T1 DELAY 0 1 0 0
C
*IYDIN 6
*B2SEQ 'ZOH BLOCK' 1 T1 DELAY 1 0
C
*B2END
C
C     **** CONNECTION OF TRANSFER FUNCTION BLOCKS COMPLETED ****
C
C     5.  **** COMPUTE CLOSED LOOP T.F. FROM D1 TO BLOCK D7 ****
      UDIN=1
      UMAGN=1.
      YDOUT=7
      PRINT*,'D1 TO D7 CLOSED LOOP T.F. WILL BE IN ZPTF20'
C     SET PRNMTRX FOR PRINTOUT OF COMPUTED STATE SPACE MATRICES
      PRNMTRX=1
*B2TF 20
C
C     6.  **** COMPUTE CLOSED LOOP TIME RESPONSE ****
      TITLE1='EXAMPLE 18, IEEE BENCHMARK NO. 3, 2-RATE MODEL'
      TITLE2='CLOSED LOOP RESPONSE BY ZTIME COMMAND'
      SAMPT=T1
      TMAGN=1.
      TEND=4.
      HRDCPY=1
      CALL ZTIME(20)
C
C     7.  **** COMPUTE CLOSED LOOP TIME RESPONSE ****
      TITLE2='CLOSED LOOP RESPONSE BY B2TIME COMMAND'
      CALL B2TIME()
C
C     8.  **** CHANGE CONNECTION DATA FOR OPEN LOOP CONFIGURATION ****
*B2INIT 'IEEE BENCHMARK PROBLEM NO.3, 2-RATE, OPEN INNER LOOP' ..
      'OLD' 2 7 1 !  NOTE USE OF 'OLD' FOR CHANGING DATA
*IYDIN -1 -3
*B2DEQ 'SUMMER BLOCK' 6 0 T1 DELAY 0 0 2 0
*B2END
C
C     **** CONNECTION OF TRANSFER FUNCTION BLOCKS COMPLETED ****
C
C     9.  **** COMPUTE OPEN LOOP TRANSFER FUNCTION ****
      UDIN=6
      UMAGN=1.
      YDOUT=2
      PRINT*,'OPEN INNER LOOP T.F. WILL BE IN ZPTF21'
*B2TF 21
C
C     10.  **** COMPUTE OPEN LOOP FREQUENCY RESPONSE ****
      NOMEGA=4
*OMEGA .01 .1 1.  25.
      RAD=0
      FNICO=1
      FBODE=1
      SAMPT=T1
      CYCLE=4
      TITLE2='OPEN LOOP FREQUENCY RESPONSE WITH COMMAND ZFREQ'
      CALL ZFREQ(21)
C
C     11.  **** COMPUTE OPEN LOOP FREQUENCY RESPONSE ****
      TITLE2='OPEN LOOP FREQUENCY RESPONSE WITH COMMAND B2FREQ'
      CALL B2FREQ()
```

Since the complete output of the example is too long to be included in this report, only a portion of it will be presented. The output from Step 5 for the command B2TF is:

```
D1 TO D7 CLOSED LOOP T.F. WILL BE IN ZPTF20
```

```
IEEE BENCHMARK PROBLEM NO. 3, 2-RATE, CLOSED LOOP
NUMBER OF CONTINUOUS BLOCKS =  2
BLOCK    DESCRIPTION

C1       G1 BLOCK
C2       G2 BLOCK

BLOCK     TRANSFER FUNCTION          INPUTS FROM BLOCKS

C1            SPTF1             S1
C2            SPTF2             S1

----------------------------------

NUMBER OF DISCRETE BLOCKS =  7
BLOCK    DESCRIPTION

D1       G3  BLOCK
D2       G4 BLOCK
D3       G5 BLOCK
D4       G6 BLOCK
D5       G7 BLOCK
D6       SUMMER BLOCK
D7       T1 SAMPLER FOR G1 BLOCK

BLOCK    TRANSFER FUNCTION  SAMPLING PERIOD     INPUTS FROM BLOCKS

D1         ZPTF3             .20000E-01   HAS NO INPUT
D2         ZPTF4             .20000E-01   D4
D3         ZPTF5             .20000E-01   D5
D4         ZPTF6             .50000E-02   C1
D5         ZPTF7             .50000E-02   C2
D6         ZPTF0             .20000E-01  -D1   , D2   ,-D3
D7         ZPTF0             .20000E-01   C1

----------------------------------

NUMBER OF SAMPLE-HOLD BLOCKS =  1
BLOCK    DESCRIPTION

S1       ZOH BLOCK

BLOCK     SAMPLING PERIOD          INPUTS FROM BLOCKS

S1           .20000E-01       D6

----------------------------------
```

```
INPUT IS CONNECTED TO BLOCK D1
MAGNITUDE OF U =  .10000E+01
BLOCK D7   IS THE OUTPUT

**********************************************************************
*    B2TF - BLOCK2: FIND TRANSFER FUNCTION FROM D1   TO D7           *
*                   AND STORE IN ZPTF20                              *
**********************************************************************

BASIC SAMPLING PERIOD COMPUTED =  .50000E-02
LCM SAMPLING PERIOD COMPUTED =  .20000E-01
TOTAL NUMBER OF TIME SEGMENTS COMPUTED =     4
****************************************************************************
*  SEGMENT   START                                                         *
*  NUMBER    TIME    DT(1)   DT(2)   DT(3)   DT(4)   DT(5)   DT(6)   DT(7) *
*                    ST(1)                                                 *
****************************************************************************
    1    .00000E+00   ON      ON      ON      ON      ON      ON      ON
                      ON
    2    .50000E-02   OFF     OFF     OFF     ON      ON      OFF     OFF
                      OFF
    3    .10000E-01   OFF     OFF     OFF     ON      ON      OFF     OFF
                      OFF
    4    .15000E-01   OFF     OFF     OFF     ON      ON      OFF     OFF
                      OFF

COMPUTED STATE SPACE REPRESENTATION IS:

   X(K+1) = PSI * X(K) +  B * U(K)
     Y(K) =   C * X(K) +  D * U(K)

WHERE X( 1: 4) ARE THE CONTINUOUS STATES
      X( 5: 9) ARE THE DISCRETE STATES
      X(10:10) ARE THE SAMPLE-HOLD STATES
      Y IS THE OUTPUT VECTOR OF LENGTH 7
      C = [FD CD HD] AND THE LCM SAMPLING PERIOD =     .20000000E-01

(THE I-TH ELEMENT OF Y IS THE OUTPUT OF THE I-TH DISCRETE CONNECTION BLOCK)

MATRIX PSI        IS
( 1, 1) .80827203E+00 ( 1, 2) .18830268E-01 ( 1, 3) .10744690E-01
( 1, 5) .28677744E+00 ( 1, 6)-.28677744E+00 ( 1, 7) .28677744E+00
( 1, 8)-.48752165E+00 ( 1, 9) .29681465E+00 ( 2, 1)-.35437644E+00
( 2, 2) .99750203E+00 ( 2, 3) .15442096E-01 ( 2, 5) .41215193E+00
( 2, 6)-.41215193E+00 ( 2, 7) .41215193E+00 ( 2, 8)-.70065829E+00
( 2, 9) .42657725E+00 ( 3, 1)-.40855435E-04 ( 3, 3) .88452617E+00
( 3, 4) .18830268E-01 ( 3, 5) .15366118E-03 ( 3, 6)-.15366118E-03
( 3, 7) .15366118E-03 ( 3, 8)-.26122401E-03 ( 3, 9) .15903932E-03
( 4, 1) .47884564E-01 ( 4, 3)-.25154124E+00 ( 4, 4) .99750203E+00
( 4, 5)-.18009841E+00 ( 4, 6) .18009841E+00 ( 4, 7)-.18009841E+00
( 4, 8) .30616729E+00 ( 4, 9)-.18640185E+00 ( 5, 5) .10000000E+01
( 6, 1)-.78195308E-01 ( 6, 6) .66670000E+00 ( 6, 8)-.49997000E+00
( 7, 3) .25327692E-02 ( 7, 7) .10000000E+01 ( 7, 9) .69966000E-01
( 8, 1) .59335504E+00 ( 8, 2) .61576808E-02 ( 8, 3) .34937286E-02
( 8, 5) .93248155E-01 ( 8, 6)-.93248155E-01 ( 8, 7) .93248155E-01
( 8, 8) .64622220E-01 ( 8, 9) .96511840E-01 ( 9, 1)-.23568806E-04
( 9, 3) .23859806E+00 ( 9, 4) .19245406E-02 ( 9, 5) .88644523E-04
( 9, 6)-.88644523E-04 ( 9, 7) .88644523E-04 ( 9, 8)-.15069569E-03
( 9, 9) .74077108E+00 (10, 1) .26588000E+00 (10, 3)-.37467000E-01
(10, 5)-.10000000E+01 (10, 6) .10000000E+01 (10, 7)-.10000000E+01
(10, 8) .17000000E+01 (10, 9)-.10350000E+01
MATRIX B          IS
( 1, 1) .75279078E-01 ( 2, 1) .10818988E+00 ( 3, 1) .40336061E-04
( 4, 1)-.47275832E-01 ( 5, 1) .24990000E-01 ( 8, 1) .24477641E-01
( 9, 1) .23269187E-04 (10, 1)-.26250000E+00
MATRIX CD         IS
( 1, 1) .10000000E+01 ( 2, 2) .10000000E+01 ( 2, 4) .17000000E+01
( 3, 3) .10000000E+01 ( 3, 5) .10350000E+01 ( 4, 4) .10000000E+01
( 5, 5) .10000000E+01 ( 6, 1)-.10000000E+01 ( 6, 2) .10000000E+01
( 6, 3)-.10000000E+01 ( 6, 4) .17000000E+01 ( 6, 5)-.10350000E+01

MATRIX FD         IS
( 2, 1) .26588000E+00 ( 3, 3) .37467000E-01 ( 4, 1) .15640000E+00
( 5, 3) .36200000E-01 ( 6, 1) .26588000E+00 ( 6, 3)-.37467000E-01
( 7, 1) .10000000E+01
MATRIX HD         IS
ALL ELEMENTS ARE ZERO
MATRIX D          IS
( 1, 1) .26250000E+00 ( 6, 1)-.26250000E+00

THE COMMON ROOTS ELIMINATED ARE

NO.        REAL            IMAG.             OMEGA            ZETA

 1      .94101122        .37660537E-01    .94176453        -.99920011
 2      .94101122       -.37660537E-01    .94176453        -.99920011
 3     1.0000000        0.
 4     0.               0.

                    THE NUMERATOR ROOTS OF ZROOT20  ARE

NO.        REAL            IMAG.             OMEGA            ZETA

 1      .66670000       0.
 2      .22314408       0.
 3      .74067933       0.
```

```
4        .90480000        0.
5        .97043947        0.

                           LOW ORDER NON-ZERO COEFFICIENT =     .64599038E-02

                           THE DENOMINATOR ROOTS OF ZROOT20  ARE

NO.         REAL             IMAG.             OMEGA             ZETA

1        .31254440         .43951523        .53931219        -.57952407
2        .31254440        -.43951523        .53931219        -.57952407
3        .98029993         .39467762E-01    .98109411        -.99919051
4        .98029993        -.39467762E-01    .98109411        -.99919051
5        .73944478        0.
6        .95273968        0.

                           LOW ORDER NON-ZERO COEFFICIENT =    -.17493088

DEGREE OF NUMERATOR OF ZPTF20  IS  5    (COEFFICIENTS IN ASCENDING ORDER)
.006459903823865 -.06115671029628 .2040925049616 -.3167094636213 .2340677016674
-.0667665525175

DEGREE OF DENOMINATOR OF ZPTF20  IS  6  (COEFFICIENTS IN ASCENDING ORDER)
-.1749308767793 1.152441778427 -3.556113404879 6.37552997883 -6.704154278874
3.794133091762 -.8869204367171
```

The above output for the command B2TF summarizes all of the connection blocks so that the analyst can easily verify that the system is correctly modeled. From the sampling periods entered for each $\mathbf{D}_i$ and $\mathbf{S}_i$ block, the basic sampling period and the LCM sampling period are computed and printed out. The discrete time segment table used in computing the transition matrix is printed out to show the analyst which sampled blocks are on or off at each of the discrete times. The computed state space matrices are printed out since the parameter PRNMTRX was set to 1. Four continuous states, five discrete states, and one sample-hold state were generated. The output of $\mathbf{C}_1$ and $\mathbf{C}_2$ are X(1) and X(3), respectively. The output of each $\mathbf{D}_i$ block is computed using matrices C and D. For example:

- The output of discrete connection block $\mathbf{D}_3$ is the sum of CD(3,3)*X(7) and CD(3,5)*X(9). The second term is due to the direct connection from $\mathbf{D}_5$ to $\mathbf{D}_3$ since the order of the numerator and denominator of G_5 are equal.
- The output of discrete connection block $\mathbf{D}_7$ is FD(7,1)*X(1) which is is the sampled output of the continuous state X(1).

The computed closed loop transfer function is stored in $\mathbf{SPTF}_{20}$. This transfer function is used in Step 6 to compute the step response by recursive evaluation of a difference equation. The response at every .5 sec is

Time	c(t)
.00000	.00000
.50000	.88987
1.0000	1.2651
1.5000	1.1570
2.0000	.92905
2.5000	.81584
3.0000	.82736
3.5000	.87830
4.0000	.90761
final value	.89170

which is identical to the results given in Ref. 5.

The low resolution printer plot is not presented. However, the high resolution plot is presented in Appendix H.

The step response computed by the state space method in Step 7 is identical to the results in Step 6. The computed state space matrices are the same as the ones computed in Step 5.

After the loop is opened in Step 8, the command B2TF is used to compute the open loop transfer function. Since the format of the output is the same as above, only the final result of this open loop transfer function is presented next:

```
INPUT IS CONNECTED TO BLOCK D6
MAGNITUDE OF U = -.10000E+01
BLOCK D2   IS THE OUTPUT

*********************************************************************
*    B2TF - BLOCK2: FIND TRANSFER FUNCTION FROM D6   TO D2          *
*                   AND STORE IN ZPTF21                             *
*********************************************************************

BASIC SAMPLING PERIOD COMPUTED =  .50000E-02
FUNDAMENTAL SAMPLING (LCM) PERIOD COMPUTED =  .20000E-01
TOTAL NUMBER OF TIME SEGMENTS COMPUTED =     4
***************************************************************************
*  SEGMENT   START                                                        *
*  NUMBER    TIME      DT(1)   DT(2)   DT(3)   DT(4)   DT(5)   DT(6)   DT(7) *
*                      ST(1)                                              *
***************************************************************************
    1     .00000E+00   ON      ON      ON      ON      ON      ON      ON
                       ON
    2     .50000E-02   OFF     OFF     OFF     ON      ON      OFF     OFF
                       OFF
    3     .10000E-01   OFF     OFF     OFF     ON      ON      OFF     OFF
                       OFF
    4     .15000E-01   OFF     OFF     OFF     ON      ON      OFF     OFF
```

```
                     OFF

COMPUTED STATE SPACE REPRESENTATION IS:

   X(K+1) = PSI * X(K) +  B * U(K)
     Y(K) =    C * X(K) +  D * U(K)

WHERE X( 1: 4) ARE THE CONTINUOUS STATES
      X( 5: 9) ARE THE DISCRETE STATES
      X(10:10) ARE THE SAMPLE-HOLD STATES
      Y IS THE OUTPUT VECTOR OF LENGTH 7
      C = [FD CD HD] AND THE LCM SAMPLING PERIOD =    .20000000E-01

(THE I-TH ELEMENT OF Y IS THE OUTPUT OF THE I-TH DISCRETE CONNECTION BLOCK)

MATRIX PSI         IS
( 1, 1) .88452042E+00 ( 1, 2) .18830268E-01 ( 1, 3) .10744690E-01
( 1, 5) .28677744E+00 ( 1, 7) .28677744E+00 ( 1, 9) .29681465E+00
( 2, 1)-.24479349E+00 ( 2, 2) .99750203E+00 ( 2, 3) .15442096E-01
( 2, 5) .41215193E+00 ( 2, 7) .41215193E+00 ( 2, 9) .42657725E+00
( 3, 3) .88452617E+00 ( 3, 4) .18830268E-01 ( 3, 5) .15366118E-03
( 3, 7) .15366118E-03 ( 3, 9) .15903932E-03 ( 4, 3)-.25154124E+00
( 4, 4) .99750203E+00 ( 4, 5)-.18009841E+00 ( 4, 7)-.18009841E+00
( 4, 9)-.18640185E+00 ( 5, 5) .10000000E+01 ( 6, 1)-.78195308E-01
( 6, 6) .66670000E+00 ( 6, 8)-.49997000E+00 ( 7, 3) .25327692E-02
( 7, 7) .10000000E+01 ( 7, 9) .69966000E-01 ( 8, 1) .61814786E+00
( 8, 2) .61576808E-02 ( 8, 3) .34937286E-02 ( 8, 5) .93248155E-01
( 8, 7) .93248155E-01 ( 8, 8) .22314408E+00 ( 8, 9) .96511840E-01
( 9, 3) .23859806E+00 ( 9, 4) .19245406E-02 ( 9, 5) .88644523E-04
( 9, 7) .88644523E-04 ( 9, 9) .74077108E+00 (10, 3)-.37467000E-01
(10, 5)-.10000000E+01 (10, 7)-.10000000E+01 (10, 9)-.10350000E+01

MATRIX B           IS
( 1, 1)-.28677744E+00 ( 2, 1)-.41215193E+00 ( 3, 1)-.15366118E-03
( 4, 1) .18009841E+00 ( 8, 1)-.93248155E-01 ( 9, 1)-.88644523E-04
(10, 1) .10000000E+01
MATRIX CD          IS
( 1, 1) .10000000E+01 ( 2, 2) .10000000E+01 ( 2, 4) .17000000E+01
( 3, 3) .10000000E+01 ( 3, 5) .10350000E+01 ( 4, 4) .10000000E+01
( 5, 5) .10000000E+01 ( 6, 1)-.10000000E+01 ( 6, 3)-.10000000E+01
( 6, 5)-.10350000E+01
MATRIX FD          IS
( 2, 1) .26588000E+00 ( 3, 3) .37467000E-01 ( 4, 1) .15640000E+00
( 5, 3) .36200000E-01 ( 6, 3)-.37467000E-01 ( 7, 1) .10000000E+01

MATRIX HD          IS
ALL ELEMENTS ARE ZERO
MATRIX D           IS
```

```
( 6, 1) .10000000E+01

THE COMMON ROOTS ELIMINATED ARE

NO.         REAL            IMAG.            OMEGA            ZETA

 1      .94101122       .37660537E-01    .94176453        -.99920011
 2      .94101122      -.37660537E-01    .94176453        -.99920011
 3      1.0000000      0.
 4     0.              0.

                    THE NUMERATOR ROOTS OF ZROOT21  ARE

NO.         REAL            IMAG.            OMEGA            ZETA

 1      .74067933      0.
 2     -.61403003      0.
 3      .96080000      0.
 4      .97044501      0.
 5      1.0000000      0.

                    LOW ORDER NON-ZERO COEFFICIENT =    .88298126E-01

                    THE DENOMINATOR ROOTS OF ZROOT21  ARE

NO.         REAL            IMAG.            OMEGA            ZETA

 1      .97235235       .61401444E-01    .97428909        -.99801215
 2      .97235235      -.61401444E-01    .97428909        -.99801215
 3      .66670000      0.
 4      .22314408      0.
 5      .72805531      0.
 6      .95003926      0.

                    LOW ORDER NON-ZERO COEFFICIENT =   -.86632750E-01

DEGREE OF NUMERATOR OF ZPTF21  IS  5    (COEFFICIENTS IN ASCENDING ORDER)
.08829812630519 -.2465973747648 .007921968700923 .5788772433174 -.6367224954357
.208222531877

DEGREE OF DENOMINATOR OF ZPTF21  IS  6  (COEFFICIENTS IN ASCENDING ORDER)
-.08663275047412 .9058444036621 -3.548191436178 6.954407222147 -7.340876774309
4.002355623639 -.8869204367171
```

In Step 10 the open loop frequency response is computed[1]. A portion of the output from this step is given below at 0.01, 0.1, 1.0., and 25. Hz. The low resolution printer plots are not presented. However, the high resolution plots are presented in Appendix H.

```
************************************************************
*     ZFREQ - FREQUENCY RESPONSE OF Z-PLANE TRANSFER       *
*             FUNCTION 21                                  *
************************************************************

SAMPLING PERIOD = .2000E-01
AUTOMATIC FREQUENCY MODE IF FAUTO.NE.0, FAUTO = 1.000

NOMEGA = 4.000    OMEGA = .1000E-01, .1000    , 1.000    , 25.00    ,

OMEGA      ZREAL ZIMAG    REAL     IMAGINARY      DB       PHASE   PHASE
 HZ                                                                MARGIN
.1000E-01 1.000  .001   .272E-03  -.898E-02   -40.935    -88.26    91.74
.1300E-01 1.000  .002   .460E-03  -.117E-01   -38.648    -87.74    92.26

                                 :

.9600E-01 1.000  .012   .251E-01  -.931E-01   -20.317    -74.90   105.10
.1000     1.000  .013   .273E-01  -.976E-01   -19.884    -74.40   105.60
.1240     1.000  .016   .419E-01  -.127E+00   -17.495    -71.71   108.29

                                 :

.9498      .993  .119  -.170E+01   .270E-07     4.618  -180.00      .00
1.000      .992  .125  -.166E+01   .933E-01     4.413  -183.22    -3.22
1.256      .988  .157  -.146E+01   .411E+00     3.596  -195.75   -15.75

                                 :

24.44     -.998  .071   .459E-01  -.373E-02   -26.731     -4.64   175.36
25.00    -1.000  .000   .456E-01  -.584E-15   -26.828       .00   180.00
```

The results of this benchmark problem are identical to the results given in Ref. 5.

To compare the system eigenvalues with those in Ref. 5, the system eigenvalues could have been computed with the command B2EIG. However, since the closed loop transfer function was computed in Step 5, the eigenvalues are equal to the denominator roots of $\mathbf{ZPTF}_{20}$ plus the four common roots at (.941011 ± .0376605), 1.00000 and 0.0 which were eliminated.

[1] The frequency response could also have been computed with command B2FREQ which would have yielded identical results.

9.6 Example 19 IEEE CACSD Benchmark Problem No. 3, 2-Rate Model (classical multirate transform method)

Solution of the 2-rate sampled-data model from the IEEE CACSD Benchmark Problem No. 3 [5] is presented in this example using classical multirate transform methods. This method of analysis is based on the block diagram reduction method in which fast rate transforms are converted to the slowest sampling rate. The block diagram of this problem, labeled in terms of the LCAP2 $\mathbf{SPTF}_i$ and $\mathbf{ZPTF}_i$ transfer functions, is given in Figure 9.7.

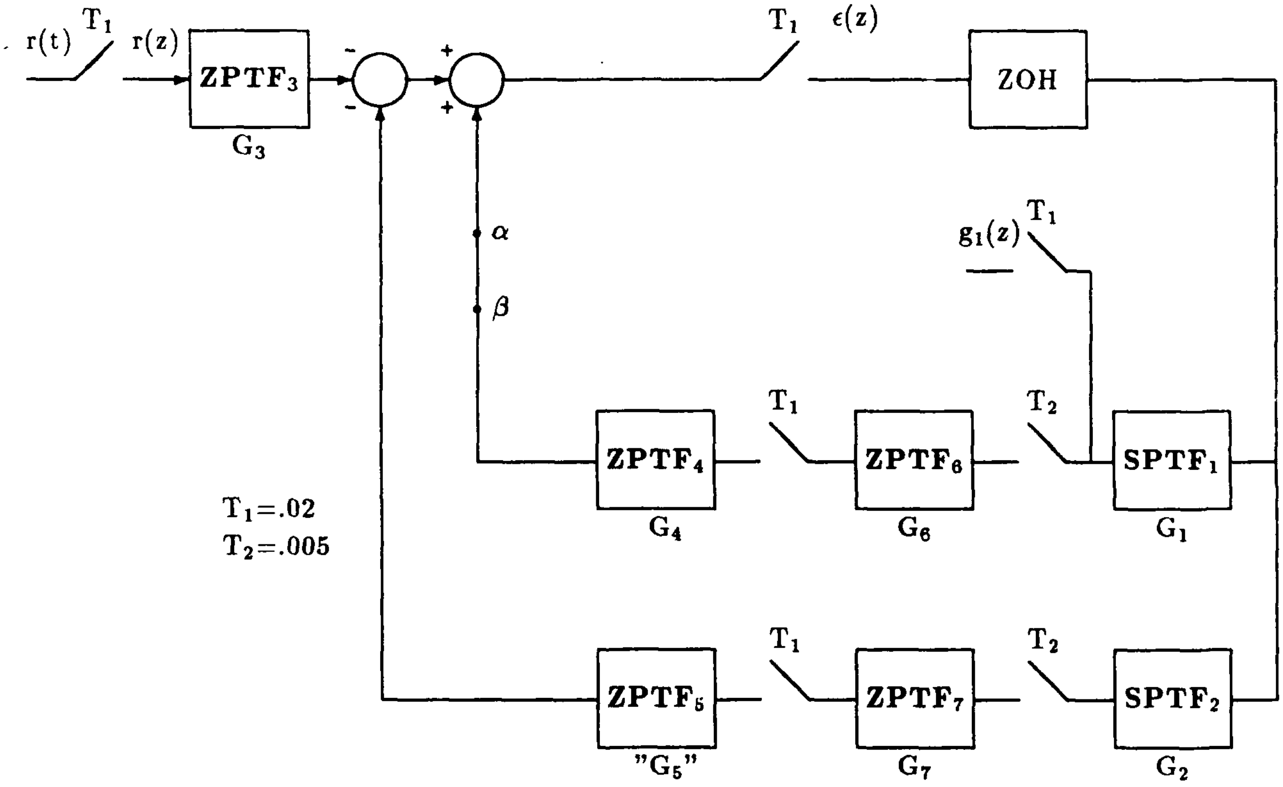


Figure 9.7: Block Diagram for IEEE Benchmark 2-Rate Model

Problem statement, system parameters, and transfer functions were previously stated in Example 18.

The first objective of this benchmark problem is to compute the time response at the output of the continuous block $G_1(s)$. Since the output is to be plotted every 0.02 sec, which is equal to the sampling period T_1, compute the z transform (at T_1) between r(k) and the sampled output of $G_1(s)$. The inverse z transform of this transfer function will yield the desired time response. Note that the system as stated in [5] does not have a sampler at the slower sampling rate at the output of $G_1(s)$. This is a fictitious or mathematical sampler used to obtain the response of a continuous variable at the slower sampling times.

The last part of this benchmark problem is to compute the open loop[1] frequency response of the system by breaking the loop between points α and β.

A change in notation for defining transforms with different sampling periods will be made so that it will be more explicit and also consistent with that used by LCAP2 commands. The sampling period of the slowest sampler will be denoted by T so that the z transform of a continuous transform G(s) is denoted by

$$\mathcal{Z}^T\{G(s)\} = G^T(z)$$

If G(s) is sampled at a rate n (an integer) times faster, the faster rate z transform of G(s) is denoted by

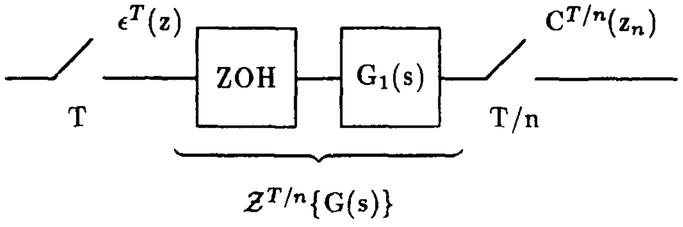
$$\mathcal{Z}^{T/n}\{G(s)\} = G^{T/n}(z_n)$$

where $z_n = [z]^{1/n}$ since $z = e^{sT}$ and $z_n = e^{sT/n}$.

The first step in simplifying the block diagram is to compute the z transforms from $\epsilon^T(z)$ to the output of $G_1(s)$ and $G_2(s)$ at the faster sampling rate. Applying the slow-to-fast multirate relationship

$$C^{T/n}(z_n) = \mathcal{Z}^{T/n}\{G(s)\} * E^T(z)$$

to the following system



the transfer function between $\epsilon^T(z)$ and the output of $G_1(s)$ at the faster sampling rate is given by

$$G_1^{T/4}(z_4) = \mathcal{Z}^{T/4}\{\frac{1-z^{-1}}{s} G_1(s)\} = \mathcal{Z}^{T/4}\{\frac{1-z_4^{-4}}{s} G_1(s)\}$$

which can be implemented with command SZMRX. The arguments of SZMRX will be selected so that the resulting z transform will be stored in $\mathbf{ZPTF}_1$. Similarly, the transfer function from $\epsilon^T(z)$ to the output of $G_2(s)$ at the faster sampling rate will be computed and stored in $\mathbf{ZPTF}_2$. Figure 9.7 can now be redrawn in terms of slow and fast rate z transforms as shown in Figure 9.8.

The next step in simplifying the block diagram is to transform the fast rate z transforms to the slower rate. First compute the products

[1] The open loop transfer function as defined by Ref. 5 is not the usual definition of an open loop transfer function used for stability analysis. To properly read out gain and phase margins at the crossover frequencies, the negative of the open loop transfer function defined by this benchmark problem would be used instead.

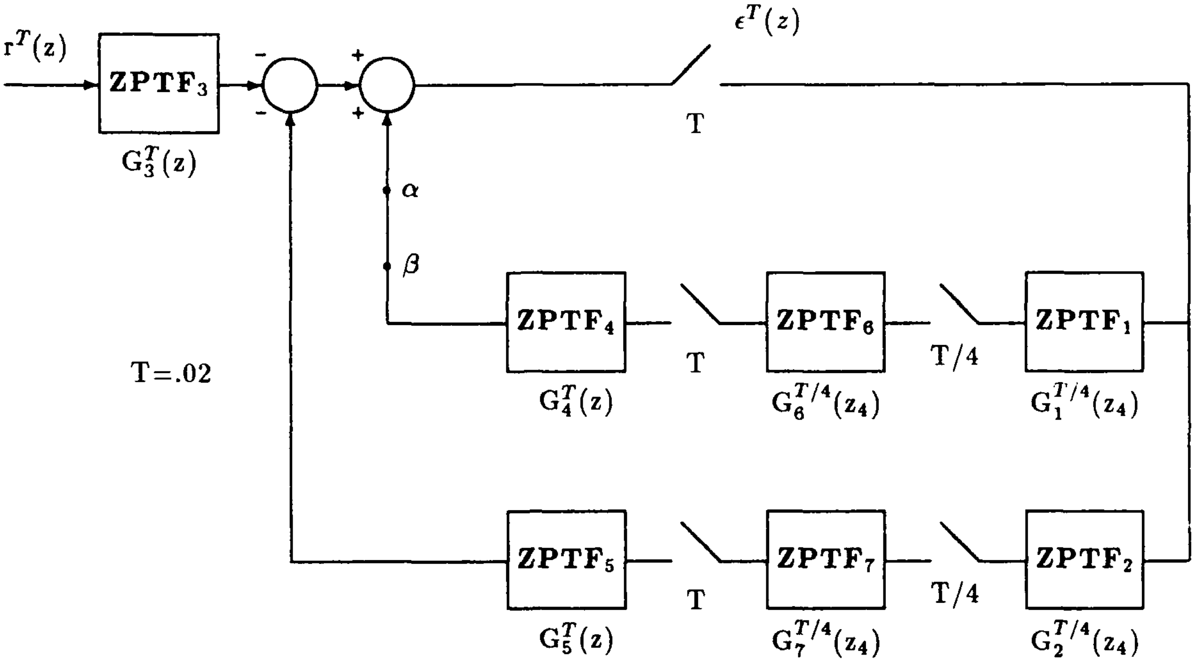


Figure 9.8: Slow and Fast Rate Z Plane Transfer Function Representation for 2-Rate Model

$$G_8^{T/4}(z_4) = G_6^{T/4}(z_4) * G_1^{T/4}(z_4)$$

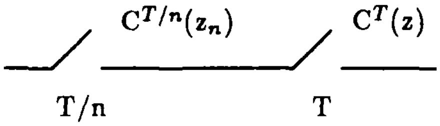
and

$$G_9^{T/4}(z_4) = G_7^{T/4}(z_4) * G_2^{T/4}(z_4)$$

and store them into **ZPTF$_8$** and **ZPTF$_9$**, respectively. Then using the fast-to-slow multirate relationship

$$C^T(z) = \frac{1}{n}\sum_{k=0}^{n-1} C^{T/n}(z_n e^{\frac{j2\pi * k}{n}})$$

which describes the following system



the transfer function between $\epsilon^T(z)$ and the output of $G_8^{T/4}(z_4)$ at the slower sampling rate is given by

$$G_{10}^T(z) = \frac{1}{4}\sum_{k=0}^{3} G_8^{T/4}(z_4 e^{\frac{j2\pi * k}{4}}) * \epsilon^T(z)$$

which can be computed with command ZMRXFM. The arguments of ZMRXFM will be selected so that the resultant z transform will be stored in $\mathbf{ZPTF}_{10}$. Similarly, the transfer function between ϵ^T(z) and the output of $G_9^{T/4}(z_4)$ at the slower sampling rate is computed and stored in $\mathbf{ZPTF}_{11}$. The block diagram can now be simplified to the single rate system in Figure 9.9.

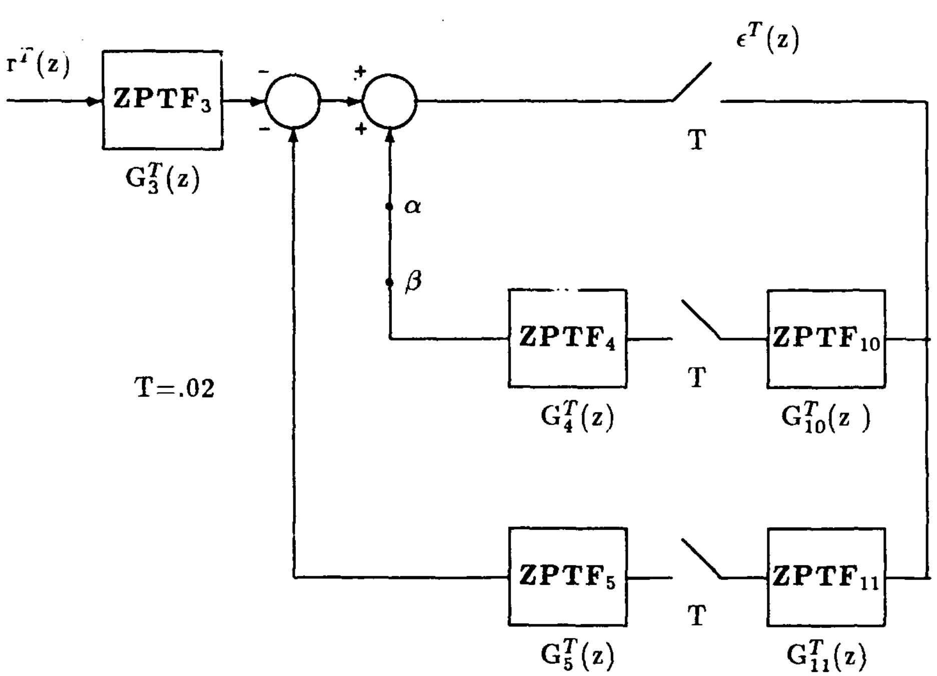


Figure 9.9: Slow Rate Z Plane Transfer Function Representation for 2-Rate Model

By inspection from Figure 9.9, the open loop transfer function with the inner loop broken between α and β, as defined in [5], is

$$\frac{G_4^T(z) * G_{10}^T}{1 + G_5^T * G_{11}^T}$$

The closed loop transfer function from $r^T(z)$ to the output of $G_1(s)$ sampled at the slower rate is not readily determined from Figure 9.9 since the output does not appear explicitly. The desired closed loop transfer function, though, can be computed as

$$\frac{G_1^T(z)}{R^T(z)} = \frac{\epsilon^T(z)}{R^T(z)} * Z^T\{\frac{1 - z^{-1}}{s} G_1(s)\}$$

where

$$\frac{\epsilon^T(z)}{R_T(z)} = \frac{-G_3^T(z)}{1 + G_5^T(z) * G_{11}^T(z) - G_4^T(z) * G_{10}^T(z)}$$

Since there are many steps involved in the solution of this problem, the PRECMP precompiler will be used. The PRECMP code is:

```
C      IEEE CACSD BENCHMARK PROBLEM NO. 3, 2-RATE MODEL
C
C      1.  **** LOAD IN S PLANE TRANSFER FUNCTIONS ****
*POLYN 1 -15*1.5 -15
*POLYD 2 13 6 1
*SPLDC 1
C
*POLYN 1 9 -.1
*POLYD 2 13 6 1
*SPLDC 2
C
C      2.  **** LOAD IN Z PLANE TRANSFER FUNCTIONS ****
*POLYN 1 -.2625*.9048 .2625
*POLYD 1 -1 1
*ZPLDC 3
C
*POLYN 1 -1.7*.9608 1.7
*POLYD 1 -.6667 1
*ZPLDC 4
C
*POLYN 1 -1.035*.9324 1.035
*POLYD 1 -1 1
*ZPLDC 5
C
*POLYN 1 .1564 .1564
*POLYD 1 -.6873 1
*ZPLDC 6
C
*POLYN 1 .0362 .0362
*POLYD 1 -.9277 1
*ZPLDC 7
C
C      3.  **** ENTER SAMPLING PERIODS ****
       T1=1/50.
       T2=1/200.
C
```

```
C      4.  *** COMPUTE SLOW-TO-FAST MULTIRATE Z TRANSFORM FROM ETA TO
C      G1(S) AT FASTER SAMPLING RATE
*SZMRX 1 1 T2 0.  1 4 !  I J SAMPT DELAY ZOH NTGER
C      NOTE USE OF COMMENT CHARACTER ! IN ABOVE STATEMENT
C
C      5.  *** COMPUTE SLOW-TO-FAST MULTIRATE Z TRANSFORM FROM ETA TO
C      G2(S) AT FASTER SAMPLING RATE
*SZMRX 2 2 T2 0.  1 4 !  I J SAMPT DELAY ZOH NTGER
C
C      6.  *** COMPUTE ROOTS OF ZPTF6 AND ZPTF7 SO THAT IN STEPS 6 AND 7
C      MULTIPLICATION WILL BE MORE ACCURATE (ROOT DATA INSTEAD OF
C      COEFFICIENT DATA WILL BE USED)
*ZPRTS 6
C
C      7.  *** COMPUTE PRODUCT OF ZPTF6 AND ZPTF1 AT FASTER RATE
*ZPMPY 8 6 1
C
C      8.  *** COMPUTE PRODUCT OF ZPTF7 AND ZPTF2 AT FASTER RATE
*ZPMPY 9 7 2
C
C      9.  *** COMPUTE FAST-TO-SLOW TRANSFER FUNCTION
*ZMRXFM 10 8 T1 4 !  I J SAMPT NTGER
*ZMRXFM 11 9 T1 4 !  I J SAMPT NTGER
C
C      10.  *** COMPUTE INNER LOOP TRANSFER FUNCTION
*ZPRTS 4 !  COMPUTE ROOTS OF ZPTF4 FOR USE WITH NEXT COMMAND
*ZPMPY 20 4 10 !  SAVE IN ZPTF20
C
C      11.  *** COMPUTE OUTER LOOP TRANSFER FUNCTION
*ZPRTS 5 !  COMPUTE ROOTS OF ZPTF5 FOR USE IN NEXT COMMAND
*ZPMPY 21 5 11 !  SAVE IN ZPTF21
C
C      12.  *** COMPUTE OPEN LOOP TRANSFER FUNCTION WITH LOOP OPENED
C      AT P1.
*ZPADD 22 0 21 !  DENOMINATOR , (ZPTF0=UNITY TRANSFER FUNCTION)
*ZPDIV 23 20 22 !  OPEN LOOP TRANSFER FUNCTION AS DEFINED BY BENCHMARK
*ZELCR 23 !  ELIMINATE COMMON ROOTS
C
C      13.  *** COMPUTE CLOSED LOOP TRANSFER FUNCTION
*ZPRTS 3
*ZPSUB 24 22 20 !  DENOMINATOR FOR ETA/R TRANSFER FUNCTION
*ZPDIV 25 3 24 !  OPEN LOOP TRANSFER FUNCTION AS DEFINED BY BENCHMARK
*ROOTN -1
*ROOTD 1
*ZPLDR 15 !  -1 TRANSFER FUNCTION
*ZPMPY 25 25 15
*SZXFM 26 1 T1 0.  1 !  I J SAMPT DELAY ZOH
C      ZPTF26 IS Z TRANSFORM OF G1(S) AT SLOWER RATE
```

```
*ZPMPY 27 25 26 !  G1(Z)/R(Z) TRANSFER FUNCTION
*ZELCR 27 !  ELIMINATE COMMON ROOTS
C
C      14.  **** COMPUTE CLOSED LOOP TIME RESPONSE ****
       TITLE1='EXAMPLE 19, IEEE BENCHMARK PROBLEM NO. 3, 2-RATE MODEL'
       TITLE2='CLOSED LOOP RESPONSE'
       SAMPT=T1
       TMAGN=1.
       TEND=4.
       HRDCPY=1
       CALL ZTIME(27)
C
C      15.  **** COMPUTE OPEN LOOP FREQUENCY RESPONSE ****
       NOMEGA=4
*OMEGA .1 1.  10.  100.
       RAD=0
       FBODE=1
       TITLE2='OPEN LOOP FREQUENCY RESPONSE'
       CALL ZFREQ(23)
```

Since the complete output of the example is too long to be included in this report, only a portion of it will be presented. The output from the last command in Step 13 (ZELCR) for the closed loop transfer function is:

```
                    THE NUMERATOR ROOTS OF ZROOT27  ARE

NO.        REAL            IMAG.            OMEGA            ZETA

 1      .94101122      -.37660537E-01    .94176453       -.99920011
 2      .94101122       .37660537E-01    .94176453       -.99920011
 3      .90480000      0.
 4      1.0000000      0.
 5      .74067933      0.
 6      .66670000      0.
 7      .22314408      0.
 8      .97043947      0.

                    LOW ORDER NON-ZERO COEFFICIENT =     .27909098

                    THE DENOMINATOR ROOTS OF ZROOT27  ARE

NO.        REAL            IMAG.            OMEGA            ZETA

 1      .98029993      -.39467762E-01    .98109411       -.99919051
 2      .98029993        .39467762E-01   .98109411        -.99919051
 3      .31254440       -.43951523       .53931219        -.57952407
 4      .31254440        .43951523       .53931219        -.57952407
 5      .94101122       -.37660537E-01   .94176453        -.99920011
 6      .94101122        .37660537E-01   .94176453        -.99920011
 7     1.0000000        0.
 8      .95273968       0.
 9      .73944478       0.

                    LOW ORDER NON-ZERO COEFFICIENT =    -7.5576404

DEGREE OF NUMERATOR OF ZPTF27  IS  8    (COEFFICIENTS IN ASCENDING ORDER)
.2790909822255 -3.513503822485 17.97327145477 -50.11146829954 84.46184663688
-88.8599113312 57.29350750668 -20.7751589087 3.252325781362

DEGREE OF DENOMINATOR OF ZPTF27  IS  9  (COEFFICIENTS IN ASCENDING ORDER)
-7.557640415876 73.38437221702 -333.6370654555 925.4071244302 -1704.954098414
2136.457676897 -1801.82495184 978.8541332973 -309.3331323539 43.20358163761

************************************************************
*     ZELCR - ELIMINATE COMMON ROOTS OF Z-PLANE             *
*             TRANSFER FUNCTION 27                          *
************************************************************

                    THE COMMON ROOTS ELIMINATED ARE
NO.        REAL             IMAG.             OMEGA            ZETA

 1      .94101122       -.37660537E-01
 2      .94101122        .37660537E-01
 3     1.0000000        0.

                    THE NUMERATOR ROOTS OF ZROOT27  ARE

NO.        REAL             IMAG.             OMEGA            ZETA

 1      .90480000       0.
 2      .74067933       0.
 3      .66670000       0.
 4      .22314408       0.
 5      .97043947       0.

                    LOW ORDER NON-ZERO COEFFICIENT =     .27909098

                    THE DENOMINATOR ROOTS OF ZROOT27  ARE

NO.        REAL             IMAG.             OMEGA            ZETA

 1      .98029993       -.39467762E-01   .98109411        -.99919051
 2      .98029993         .39467762E-01   .98109411        -.99919051
 3      .31254440        -.43951523       .53931219        -.57952407
 4      .31254440         .43951523       .53931219        -.57952407
 5      .95273968        0.
 6      .73944478        0.

                        LOW ORDER NON-ZERO COEFFICIENT =   -7.5576404

DEGREE OF NUMERATOR OF ZPTF27  IS  5    (COEFFICIENTS IN ASCENDING ORDER)
.2790909822255 -2.642188925974 8.817527199738 -13.68298316698 10.11256305773
-2.884554202357

DEGREE OF DENOMINATOR OF ZPTF27  IS  6  (COEFFICIENTS IN ASCENDING ORDER)
-7.557640415876 49.78961245687 -153.6368358003 275.4457299234 -289.6434766984
163.9201387739 -38.31813949378
```

This closed loop transfer function is identical to the one computed in Example 18 ($\mathbf{ZPTF}_{20}$) using the Kalman-Bertram method. The roots agree to all eight significant digits. The coefficients are not the same since they are normalized differently.[1]

Since the time response from step 14 is identical (to 5 significant digits) to the response computed in Example 18, it will not be repeated here.

The output from Step 12 for the open loop transfer function is:

```
                        THE NUMERATOR ROOTS OF ZROOT23  ARE

NO.         REAL            IMAG.            OMEGA             ZETA

 1      .94101122        -.37660537F-01   .94176453        -.99920011
 2      .94101122         .37660537E-01   .94176453        -.99920011
 3      .96080000        0.
 4     -.61403003        0.
 5      .97044501        0.
 6     1.0000000         0.
 7      .74067933        0.

                        LOW ORDER NON-ZERO COEFFICIENT =   -22.604320

                        THE DENOMINATOR ROOTS OF ZROOT23  ARE

NO.         REAL            IMAG.            OMEGA             ZETA
```

[1] They can be normalized to match $\mathbf{ZPTF}_{20}$ in Example 18 by using command ZNORM with NRMHI=1. and KNORM=-.8869204367171

```
  1      .94101122       -.37660537E-01    .94176453       -.99920011
  2      .94101122        .37660537E-01    .94176453       -.99920011
  3      .97235235       -.61401444E-01    .97428909       -.99801215
  4      .97235235        .61401444E-01    .97428909       -.99801215
  5      .66670000       0.
   6      .22314408       0.
   7      .95003926       0.
   8      .72805531       0.

                         LOW ORDER NON-ZERO COEFFICIENT =    22.177984

DEGREE OF NUMERATOR OF ZPTF23  IS  7    (COEFFICIENTS IN ASCENDING ORDER)
-22.60432033414 111.0947165917 -161.4722917927 -72.71149184799 475.1752142865
-566.2754878395 296.8948447107 -60.10118377454

DEGREE OF DENOMINATOR OF ZPTF23  IS  8  (COEFFICIENTS IN ASCENDING ORDER)
22.17798412137 -278.9572814681 1425.420341105 -3969.258307125 6681.222504636
-7019.668562258 4520.097788189 -1637.034447198256.

************************************************************
*     ZELCR - ELIMINATE COMMON ROOTS OF Z-PLANE            *
*             TRANSFER FUNCTION 23                         *
************************************************************

                          THE COMMON ROOTS ELIMINATED ARE

NO.        REAL              IMAG.             OMEGA            ZETA

  1      .94101122       -.37660537E-01
  2      .94101122        .37660537E-01

                         THE NUMERATOR ROOTS OF ZROOT23  ARE
NO.        REAL              IMAG.             OMEGA            ZETA

  1      .96080000       0.
  2     -.61403003       0.
  3      .97044501       0.
  4     1.0000000        0.
  5      .74067933       0.

                         LOW ORDER NON-ZERO COEFFICIENT =   -22.604320

                         THE DENOMINATOR ROOTS OF ZROOT23  ARE

NO.        REAL              IMAG.             OMEGA            ZETA

  1      .97235235       -.61401444E-01    .97428909       -.99801215
```

```
2       .97235235        .61401444E-01    .97428909        -.99801215
3       .66670000       0.
4       .22314408       0.
5       .95003926       0.
6       .72805531       C.

                      LOW ORDER NON-ZERO COEFFICIENT =     22.177984

DEGREE OF NUMERATOR OF ZPTF23  IS  5    (COEFFICIENTS IN ASCENDING ORDER)
-22.60432033414 63.12892793987 -2.028023987502 -148.1925742894 163.0009588318
-53.30496816062

DEGREE OF DENOMINATOR OF ZPTF23  IS  6  (COEFFICIENTS IN ASCENDING ORDER)
22.17798412137 -231.8961673374 908.3370076613 -1780.328248869 1879.264454223
-1024.603039651 227.0516317996
```

This open loop transfer function is identical to the one computed in Example 18 ($\mathbf{ZPTF}_{21}$) using the Kalman-Bertram method. The roots agree to all eight significant digits. The coefficients are not the same since they are normalized differently.[1]

Since the frequency response from step 15 is identical to the response computed in Example 18, it will not be repeated here.

Although the classical transform method for analyzing this multirate benchmark problem requires more effort to set up, it provides some features not offered by the automated method used in Example 18.

- The commands utilized in performing the block diagram reduction, particularly when converting from a fast rate transform to a slow rate transform, enables the analyst to "track" s plane roots to the z plane at different sampling sampling rates. With the Kalman-Bertram method employed by the automated method, the z plane roots are more difficult to correlated with s plane roots.
- Time delays in the s plane can be easily modeled with classical transform methods. At the present time the automated method cannot handle time delays

For this problem, block diagram reduction was performed entirely in the z plane without loss of computational accuracy. For higher order systems, block diagram operations involving addition or subtraction of z plane transfer functions will result in computational error. For that case the block diagram reduction should be performed in the w plane instead. In the files which contain this benchmark problem, the commands for the w plane solution are also included to demonstrate how a w plane analysis is set up.

As stated in Chapter 7, the classical transform method cannot always be applied in analyzing a multirate system. There can be configurations in which this method cannot be used. The automated analysis method based on the Kalman-Bertram method, on the other hand, is a general analysis

[1] They can be normalized to match $\mathbf{ZPTF}_{21}$ in Example 18 by using command ZNORM with NRMHI=1. and KNORM=-.8869204367171

method which can always be used for any multirate system in which the ratios of the sampling rates are integers[1]. In practice, multirate systems which are modeled as a connection of transfer functions should be analyzed by the Kalman-Bertram method since it is a general method and is easy to set up. Then, if applicable, the analysis can be complemented by the classical transform method since it can provide some insight to the problem not offered by the Kalman-Bertram method.

[1] The Kalman-Bertram method will yield a transition matrix at the LCM sampling period from which eigenvalues and root loci can always be computed. Transfer function evaluation using command B2TF and frequency response using command B2FREQ can only be obtained if the slowest sampling period is also equal to the LCM sampling period of the system.

9.7 Example 20 IEEE CACSD Benchmark Problem No. 3, 4-Rate Model (automated analysis method)

Solution of the 4-rate sampled-data model from the IEEE CACSD Benchmark Problem No. 3 [5] is presented in this example using the LCAP2 automated analysis method based on transfer function connection blocks. This automated method will show the ease in setting up the analysis to solve for the objectives of the benchmark problem. The additional setup effort of this problem over the 2-rate problem is minimal relative to the added complexity of the problem[1]. The block diagram of this problem, labeled in terms of (1) LCAP2 transfer function connection blocks and (2) LCAP2 $\mathbf{SPTF}_i$ and $\mathbf{ZPTF}_i$ transfer functions, is given in Figure 9.10. This figure includes a unity discrete connection block $\mathbf{D}_{13}$ used to define the sampled output of G_1 at the slower sampling rate and a unity discrete connection block $\mathbf{D}_6$ to be used in defining an open loop transfer function.

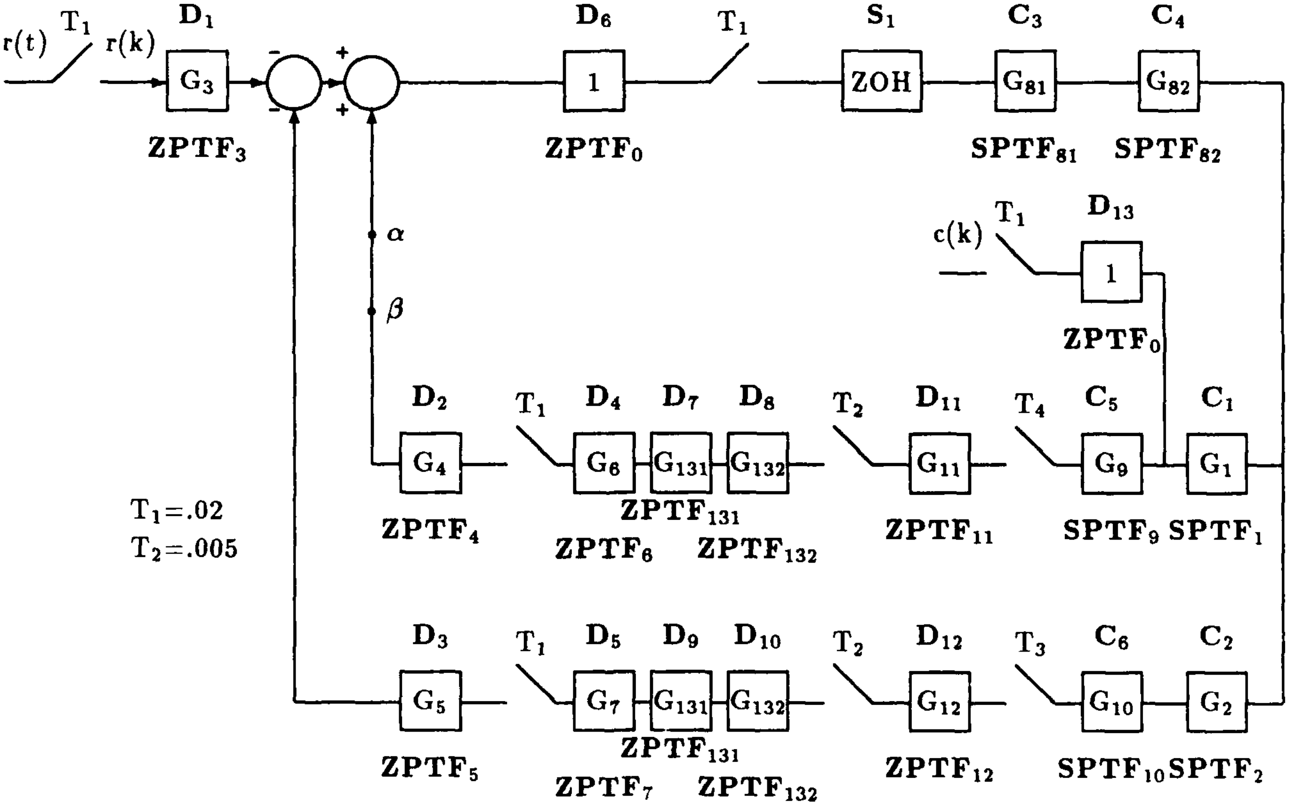


Figure 9.10: Block Diagram for IEEE Benchmark 4-Rate Model

Problem statement, system parameters and transfer functions are the same as those given in Example 18, plus the following:

```
T3=1/1000; T4=1/2000
***continuous transfer functions
G8 = G81*G82
```

[1] This would not be true if the classical transform method of analysis were used instead. See Example 21.

$G_{81} = 62.8*62.8 / (s**2 + 94.25*s + 62.8*62.8)$

$G_{82} = 1 / (0.031*s + 1)$

$G_{9} = G_{91}*G_{92}$

$G_{91} = 200*200 / (s**2 + 2*0.125*200*s + 200*200)$

$G_{92} = 1 / s$

$G_{10} = G_{101}*G_{102}$

$G_{101} = 100*100 / (s**2 + 2*0.125*100*s + 100*100)$

$G_{102} = 1 / s$

***discrete transfer functions

$G_{11} = G_{111}*G_{112}*G_{113}$

$G_{111} = (z - 1) / z$

$G_{112} = 0.9942*(z**2 - 1.6179*z + 1) / (z**2 - 1/6084*z + 0.9883)$

$G_{113} = 1 / (z - 1)$

$G_{12} = 1 / z$

$G_{13} = G_{131}*G_{132}$

$G_{131} = (z - 1) / (T_2*z)$

$G_{132} = 0.9994*(z**2 - 1.786*z + 1) / (z**2 - 1.785*z + 0.9988)$

$G_{14} = G_{13}$

The same 10 steps described in Examples 18 will be used to solve this 4-rate problem. An additional 11-th step using command B2FREQ will be used to obtain the open loop frequency response by state space methods so that its results can be compared with those obtained from the rational transfer function in Step 10.

As with Example 18, the PRECMP precompiler will be used to simplify the setup of this problem. The PRECMP code is:

```
C      IEEE CACSD BENCHMARK PROBLEM NO. 3, 4-RATE MODEL
C
C      1.  **** LOAD IN S PLANE TRANSFER FUNCTIONS ****
*POLYN 1 -15*1.5 -15
*POLYD 2 13 6 1
*SPLDC 1
C
*POLYN 1 9 -.1
*POLYN 1 9 -.1
*POLYD 2 13 6 1
*SPLDC 2
C
*POLYN 0 200.*200.
*POLYD 3 0.  200.*200.  2.*.125*200.  1.
*SPLDC 9
C
*POLYN 0 100.*100.
*POLYD 3 0.  100.*100.  2.*.125*100.  1.
```

```
*SPLDC 10
C
*POLYN 0 62.8*62.8
*POLYD 2 62.8*62.8 94.25 1
*SPLDC 81
C
*POLYN 0 1
*POLYD 1 1 .03
*SPLDC 82
C
C      2.  **** ENTER SAMPLING PERIODS ****
       T1=1/50.
       T2=1/200.
       T3=1/1000.
       T4=1/2000.
C
C      3.  **** LOAD IN Z PLANE TRANSFER FUNCTIONS ****
*POLYN 1 -.2625*.9048 .2625
*POLYD 1 -1 1
*ZPLDC 3
C
*POLYN 1 -1.7*.9608 1.7
*POLYD 1 -.6667 1
*ZPLDC 4
C
*POLYN 1 -1.035*.9324 1.035
*POLYD 1 -1 1
*ZPLDC 5
C
*POLYN 1 .1564 .1564
*POLYD 1 -.6873 1.0
*ZPLDC 6
C
*POLYN 1 .0362 .0362
*POLYD 1 -.9277 1
*ZPLDC 7
C
*POLYN 2 .9942 -.9942*1.6179 .9942
*POLYD 3 0.  .9883 -1.6084 1.
*ZPLDC 11
C
*POLYN 0 1
*POLYD 1 0 1
*ZPLDC 12
C
*POLYN 0 1
*POLYD 0 1
*ZPLDC 15
```

```
C
*POLYN 1 -1 1
*POLYD 1 0 T2
*ZPLDC 131
C
*POLYN 2 .9994 -.9994*1.786 .9994
*POLYD 2 .9988 -1.785 1.
*ZPLDC 132
C
C      4.  **** DEFINE AND CONNECT TRANSFER FUNCTION BLOCKS FOR CLOSED
C          **** LOOP CONFIGURATION
       DELAY=0.
C      NOTE: DELAY OPTION NOT IMPLEMENTED YET
C
C      *******************************************************
C
*B2INIT 'IEEE CACSD BENCHMARK PROBLEM NO. 3, 4-RATE MODEL' ..
       'NEW' 6 13 1
C
       DELAY=0.
C
*IYCIN 4
       INDX=1
       ISPTF=1
       NYCIN=1
       NXSIN=0
*B2CEQ 'G1 BLOCK' INDX ISPTF DELAY 0 NYCIN NXSIN
C
*IYCIN 4
*B2CEQ 'G2 BLOCK' 2 2 DELAY 0 1 0
C
*IXSIN 1
*B2CEQ 'G81 BLOCK' 3 81 DELAY 0 0 1
C
*IYCIN 3
*B2CEQ 'G82 BLOCK' 4 82 DELAY 0 1 0
C
*IYCIN 1
*B2CEQ 'G9 BLOCK' 5 9 DELAY 0 1 0
C
*IYCIN 2
*B2CEQ 'G10 BLOCK' 6 10 DELAY 0 1 0
C
*B2DEQ 'G3 BLOCK' 1 3 T1 DELAY 0 0 0 0
C
*IYDIN 4
*B2DEQ 'G4 BLOCK' 2 4 T1 DELAY 0 0 1 0
C
```

```
*IYDIN 5
*B2DEQ 'G5 BLOCK ' 3 5 T1 DELAY 0 0 1 0
C
*IYDIN 7
*B2DEQ 'G6 BLOCK' 4 6 T2 DELAY 0 0 1 0
C
*IYDIN 9
*B2DEQ 'G7 BLOCK' 5 7 T2 DELAY 0 0 1 0
C
*IYDIN -1 2 -3
*B2DEQ 'SUMMER BLOCK' 6 15 T1 DELAY 0 0 3 0
C
*IYDIN 8
*B2DEQ 'G131 BLOCK' 7 131 T2 DELAY 0 0 1 0
C
*IYDIN 11
*B2DEQ 'G132 BLOCK' 8 132 T2 DELAY 0 0 1 0
C
*IYDIN 10
*B2DEQ 'G131 BLOCK' 9 131 T2 DELAY 0 0 1 0
C
*IYDIN 12
*B2DEQ 'G132 BLOCK' 10 132 T2 DELAY 0 0 1 0
C
*IYCIN 5
*B2DEQ 'G11 BLOCK' 11 11 T4 DELAY 0 1 0 0
C
*IYCIN 6
*B2DEQ 'G12 BLOCK' 12 12 T3 DELAY 0 1 0 0
C
*IYCIN 1
*B2DEQ 'T1 SAMPLER AT BLOCK C1' 13 0 T1 DELAY 0 1 0 0
C
*IYDIN 6
*B2SEQ 'ZOH BLOCK' 1 T1 DELAY 1 0
C
C      ***************************************************
C
*B2END
C
C      **** CONNECTION OF TRANSFER FUNCTION BLOCKS COMPLETED ****
C
C      5.  **** COMPUTE CLOSED LOOP T.F. FROM D1 TO BLOCK D13 ****
       UDIN=1
       UMAGN=1.
       YDOUT=13
       PRNMTRX=1
       PRINT*,'D1 TO D13 CLOSED LOOP T.F. WILL BE IN ZPTF20'
```

```
*B2TF 20
C
C      6.  **** COMPUTE CLOSED LOOP TIME RESPONSE ****
       TITLE1='IEEE BENCHMARK PROBLEM NO. 3, 4-RATE MODEL'
       TITLE2='CLOSED LOOP RESPONSE WITH COMMAND ZTIME'
       SAMPT=T1
       TMAGN=1
       TEND=4
       HRDCPY=1
       CALL ZTIME(20)
C
C      7.  **** COMPUTE CLOSED LOOP FREQUENCY RESPONSE
       TITLE2='CLOSED LOOP RESPONSE WITH COMMAND B2TIME'
       CALL B2TIME()
C
C      8.  **** CHANGE CONNECTION DATA FOR OPEN LOOP CONFIGURATION ****
*B2INIT 'IEEE BENCHMARK NO. 3, OPEN INNER LOOP T.F.' ..
       'OLD' 6 13 1
*IYDIN -1 -3
*B2DEQ 'SUMMER BLOCK' 6 0 T1 DELAY 0 0 2 0
*B2END
C
C      **** CONNECTION OF TRANSFER FUNCTION BLOCKS COMPLETED ****
C
C      9.  **** COMPUTE OPEN LOOP TRANSFER FUNCTION ****
       UDIN=6
       UMAGN=1
       YDOUT=2
       PRINT*,'OPEN LOOP T.F. WILL BE IN ZPTF21'
*B2TF 21
C
C      10.  **** COMPUTE OPEN LOOP FREQUENCY RESPONSE ****
       NOMEGA=4
*OMEGA .01 .1 1.  25.
       RAD=0
       FNICO=1
       FBODE=1
       SAMPT=T1
       TITLE2='OPEN LOOP FREQUENCY RESPONSE WITH COMMAND ZFREQ'
       CALL ZFREQ(21)
C
C      *******************************************************************
C
C      11.  **** COMPUTE OPEN LOOP FREQUENCY RESPONSE ****
       TITLE2='OPEN LOOP FREQUENCY RESPONSE WITH COMMAND B2FREQ'
       CALL B2FREQ()
```

Since the complete output of the example is too long to be included in this report, only a portion of it will be presented. The output from Step 5 for the command B2TF is:

```
D1 TO D13 CLOSED LOOP T.F. WILL BE IN ZPTF20

IEEE CACSD BENCHMARK PROBLEM NO. 3, 4-RATE MODEL
NUMBER OF CONTINUOUS BLOCKS =  6
BLOCK    DESCRIPTION
C1       G1 BLOCK
C2       G2  BLOCK
C3       G81 BLOCK
C4       G82 BLOCK
C5       G9 BLOCK
C6       G10 BLOCK

BLOCK     TRANSFER FUNCTION          INPUTS FROM BLOCKS

C1            SPTF1             C4
C2            SPTF2             C4
C3            SPTF81            S1
C4            SPTF82            C3
C5            SPTF9             C1
C6            SPTF10            C2

----------------------------------

NUMBER OF DISCRETE BLOCKS = 13
BLOCK    DESCRIPTION
D1       G3  BLOCK
D2       G4 BLOCK
D3       G5 BLOCK
D4       G6 BLOCK
D5       G7 BLOCK
D6       SUMMER BLOCK
D7       G131 BLOCK
D8       G132 BLOCK
D9       G131 BLOCK
D10      G132 BLOCK
D11      G11 BLOCK
D12      G12 BLOCK
D13      T1 SAMPLER AT BLOCK C1

BLOCK    TRANSFER FUNCTION  SAMPLING PERIOD     INPUTS FROM BLOCKS

D1         ZPTF3              .20000E-01   HAS NO INPUT
```

```
D2          ZPTF4              .20000E-01   D4
D3          ZPTF5              .20000E-01   D5
D4          ZPTF6              .50000E-02   D7
D5          ZPTF7              .50000E-02   D9
D6          ZPTF15             .20000E-01  -D1  , D2  ,-D3
D7          ZPTF131            .50000E-02   D8
D8          ZPTF132            .50000E-02   D11
D9          ZPTF131            .50000E-02   D10
D10         ZPTF132            .50000E-02   D12
D11         ZPTF11             .50000E-03   C5
D12         ZPTF12             .10000E-02   C6
D13         ZPTF0              .20000E-01   C1

----------------------------------

NUMBER OF SAMPLE-HOLD BLOCKS =  1
BLOCK    DESCRIPTION
S1       ZOH BLOCK

BLOCK     SAMPLING PERIOD          INPUTS FROM BLOCKS

S1           .20000E-01       D6

----------------------------------

INPUT IS CONNECTED TO BLOCK D1
MAGNITUDE OF U =  .10000E+01
BLOCK D13  IS THE OUTPUT

**********************************************************************
*    B2TF - BLOCK2: FIND TRANSFER FUNCTION FROM D1   TO D13          *
*                    AND STORE IN ZPTF20                             *
**********************************************************************

BASIC SAMPLING PERIOD COMPUTED =  .50000E-03
LCM SAMPLING PERIOD COMPUTED =  .20000E-01
TOTAL NUMBER OF TIME SEGMENTS COMPUTED =    40

****************************************************************************
*  SEGMENT   START                                                         *
*  NUMBER    TIME     DT(1)   DT(2)   DT(3)   DT(4)   DT(5)   DT(6)   DT(7) *
*                     DT(8)   DT(9)   DT(10)  DT(11)  DT(12)  DT(13)  ST(1) *
****************************************************************************
    1     .00000E+00   ON      ON      ON      ON      ON      ON      ON
                       ON      ON      ON      ON      ON      ON      ON
    2     .50000E-03   OFF     OFF     OFF     OFF     OFF     OFF     OFF
                       OFF     OFF     OFF     ON      OFF     OFF     OFF
    3     .10000E-02   OFF     OFF     OFF     OFF     OFF     OFF     OFF
```

```
                      OFF    OFF    OFF    ON     ON     OFF    OFF
    4   .15000E-02    OFF    OFF    OFF    OFF    OFF    OFF    OFF
                      OFF    OFF    OFF    ON     OFF    OFF    OFF
    5   .20000E-02    OFF    OFF    OFF    OFF    OFF    OFF    OFF
                      OFF    OFF    OFF    ON     ON     OFF    OFF
    6   .25000E-02    OFF    OFF    OFF    OFF    OFF    OFF    OFF

                          :

   38    .18500E-01    OFF    OFF    OFF    OFF    OFF    OFF    OFF
                       OFF    OFF    OFF    ON     OFF    OFF    OFF
   39    .19000E-01    OFF    OFF    OFF    OFF    OFF    OFF    OFF
                       OFF    OFF    OFF    ON     ON     OFF    OFF
   40    .19500E-01    OFF    OFF    OFF    OFF    OFF    OFF    OFF
                       OFF    OFF    OFF    ON     OFF    OFF    OFF

COMPUTED STATE SPACE REPRESENTATION IS:

  X(K+1) = PSI * X(K) +  B * U(K)
    Y(K) =   C * X(K) + DD * U(K)

WHERE X( 1:13) ARE THE CONTINUOUS STATES
      X(14:28) ARE THE DISCRETE STATES
      X(29:29) ARE THE SAMPLE-HOLD STATES
      Y IS THE OUTPUT VECTOR OF LENGTH 13
      C = [FD CD HD] AND THE LCM SAMPLING PERIOD =  .20000E-01
(THE I-TH ELEMENT OF Y IS THE OUTPUT OF THE I-TH DISCRETE CONNECTION BLOCK)

MATRIX PSI         IS
( 1, 1) .88452042E+00 ( 1, 2) .18830268E-01 ( 1, 5)-.38961641E-01
( 1, 6)-.33798898E-03 ( 1, 7)-.20825888E+00 ( 1,14) .77014595E-02
( 1,15)-.77014595E-02 ( 1,16) .77014595E-02 ( 1,17)-.13092481E-01
( 1,18) .79710106E-02 ( 1,19)-.20476641E-02 ( 1,20)-.40953281E+00
( 1,22) .28855058E-03 ( 1,23) .57710117E-01 ( 1,25)-.40928709E+00
( 1,28) .57675491E-01 ( 2, 1)-.24479349E+00 ( 2, 2) .99750203E+00
( 2, 5)-.56348217E-01 ( 2, 6)-.49485848E-03 ( 2, 7)-.29782120E+00
( 2,14) .11342097E-01 ( 2,15)-.11342097E-01 ( 2,16) .11342097E-01
( 2,17)-.19281565E-01 ( 2,18) .11739070E-01 ( 2,19)-.30156368E-02
( 2,20)-.60312735E+00 ( 2,22) .42495435E-03 ( 2,23) .84990870E-01
( 2,25)-.60276548E+00 ( 2,28) .84939875E-01 ( 3, 3) .88452042E+00
( 3, 4) .18830268E-01 ( 3, 5)-.55346093E-04 ( 3, 6)-.10698617E-05
( 3, 7) .33357430E-04 ( 3,14) .30838061E-04 ( 3,15)-.30838061E-04
( 3,16) .30838061E-04 ( 3,17)-.52424704E-04 ( 3,18) .31917394E-04
( 3,19)-.81992238E-05 ( 3,20)-.16398448E-02 ( 3,22) .11554096E-05
( 3,23) .23108193E-03 ( 3,25)-.16388608E-02 ( 3,28) .23094328E-03
( 4, 3)-.24479349E+00 ( 4, 4) .99750203E+00 ( 4, 5) .24310738E-01
( 4, 6) .20819951E-03 ( 4, 7) .13145031E+00 ( 4,14)-.47145488E-02
( 4,15) .47145488E-02 ( 4,16)-.47145488E-02 ( 4,17) .80147330E-02
( 4,18)-.48795580E-02 ( 4,19) .12535042E-02 ( 4,20) .25070085E+00
( 4,22)-.17664000E-03 ( 4,23)-.35328000E-01 ( 4,25) .25055043E+00
( 4,28)-.35306803E-01 ( 5, 5)-.63581423E-01 ( 5, 6) .69281979E-02
( 5,14)-.41059877E+00 ( 5,15) .41059877E+00 ( 5,16)-.41059877E+00
( 5,17) .69801791E+00 ( 5,18)-.42496973E+00 ( 5,19) .10917000E+00
( 5,20) .21834000E+02 ( 5,22)-.15383904E-01 ( 5,23)-.30767808E+01
( 5,25) .21820900E+02 ( 5,28)-.30749347E+01 ( 6, 5)-.27323704E+02
( 6, 6) .58940123E+00 ( 6,14)-.66022638E+02 ( 6,15) .66022638E+02
( 6,16)-.66022638E+02 ( 6,17) .11223848E+03 ( 6,18)-.68333430E+02
( 6,19) .17554099E+02 ( 6,20) .35108198E+04 ( 6,22)-.24736702E+01
( 6,23)-.49473404E+03 ( 6,25) .35087133E+04 ( 6,28)-.49443720E+03
( 7, 5) .14095167E+00 ( 7, 6) .26996477E-02 ( 7, 7) .51341712E+00
( 7,14)-.91189407E-01 ( 7,15) .91189407E-01 ( 7,16)-.91189407E-01
( 7,17) .15502199E+00 ( 7,18)-.94381036E-01 ( 7,19) .24245440E-01
( 7,20) .48490879E+01 ( 7,22)-.34165935E-02 ( 7,23)-.68331870E+00
( 7,25) .48461785E+01 ( 7,28)-.68290871E+00 ( 8, 1) .19603809E-01
( 8, 2) .13395871E-03 ( 8, 5)-.19827809E-03 ( 8, 6)-.80149546E-06
( 8, 7)-.17395870E-02 ( 8, 8)-.35442209E+00 ( 8, 9)-.22494534E-02
( 8,10) .36672369E-04 ( 8,14) .10293583E-04 ( 8,15)-.10293583E-04
( 8,16) .10293583E-04 ( 8,17)-.17499091E-04 ( 8,18) .10653859E-04
( 8,19)-.27368579E-05 ( 8,20)-.54737157E-03 ( 8,22) .38566968E-06
( 8,23) .77133936E-04 ( 8,25)-.54704315E-03 ( 8,28) .77087655E-04
( 9, 1) .23277209E+01 ( 9, 2) .26301745E-01 ( 9, 5)-.46051459E-01
( 9, 6)-.23835286E-03 ( 9, 7)-.32606433E+00 ( 9, 8) .89978137E+02
( 9, 9)-.46689476E+00 ( 9,10)-.41583496E-03 ( 9,14) .36756490E-02
( 9,15)-.36756490E-02 ( 9,16) .36756490E-02 ( 9,17)-.62486033E-02
( 9,18) .38042967E-02 ( 9,19)-.97728156E-03 ( 9,20)-.19545631E+00
( 9,22) .13771554E-03 ( 9,23) .27543108E-01 ( 9,25)-.19533904E+00
( 9,28) .27526582E-01 (10, 1) .75321073E+03 (10, 2) .76860695E+01
(10, 5)-.13519559E+02 (10, 6)-.78111277E-01 (10, 7)-.94222273E+02
(10,10) .10000000E+01 (10,14) .13517689E+01 (10,15)-.13517689E+01
(10,16) .13517689E+01 (10,17)-.22980071E+01 (10,18) .13990808E+01
(10,19)-.35940831E+00 (10,20)-.71881662E+02 (10,22) .50646725E-01
(10,23) .10129345E+02 (10,25)-.71838533E+02 (10,28) .10123267E+02
(11, 3) .94395901E-02 (11, 4) .51974283E-04 (11, 5)-.31368028E-06
(11, 6)-.12614667E-08 (11, 7)-.27682709E-05 (11,11)-.40279695E+00
(11,12) .71879585E-02 (11,13) .12230980E-03 (11,14) .16164417E-07
(11,15)-.16164417E-07 (11,16) .16164417E-07 (11,17)-.27479509E-07
(11,18) .16730172E-07 (11,19)-.42977952E-08 (11,20)-.85955904E-06
(11,22) .60563221E-09 (11,23) .12112644E-06 (11,25)-.85904330E-06
(11,28) .12105377E-06 (12, 3) .14017745E+01 (12, 4) .10738947E-01
(12, 5)-.65578316E-04 (12, 6)-.34521694E-06 (12, 7)-.45312154E-03
(12,11)-.71879585E+02 (12,12)-.22309799E+00 (12,13) .10245703E-01
(12,14) .53791332E-05 (12,15)-.53791332E-05 (12,16) .53791332E-05
(12,17)-.91445265E-05 (12,18) .55674029E-05 (12,19)-.14302039E-05
(12,20)-.28604079E-03 (12,22) .20153998E-06 (12,23) .40307997E-04
(12,25)-.28586916E-03 (12,28) .40283812E-04 (13, 3) .18830268E+03
(13, 4) .19215174E+01 (13, 5)-.10698617E-01 (13, 6)-.78192983E-04
```

```
(13, 7)-.56105587E-01 (13,13) .10000000E+01 (13,14) .15231246E-02
(13,15)-.15231246E-02 (13,16) .15231246E-02 (13,17)-.25893118E-02
(13,18) .15764339E-02 (13,19)-.40496836E-03 (13,20)-.80993672E-01
(13,22) .57066908E-04 (13,23) .11413382E-01 (13,25)-.80945075E-01
(13,28) .11406534E-01 (14,14) .10000000E+01 (15,15) .66670000E+00
(15,17)-.49997000E+00 (15,19)-.78195308E-01 (15,20)-.15639062E+02
(15,25)-.15629678E+02 (16,16) .10000000E+01 (16,18) .69966000E-01
(16,22) .25327692E-02 (16,23) .50655384E+00 (16,28) .50624991E+00
(17, 1) .51530531E+00 (17, 2) .23657321E-02 (17, 5)-.28011665E-02
(17, 6)-.77639362E-05 (17, 7)-.32059235E-01 (17, 8)-.36367406E+02
(17, 9)-.53196861E-01 (17,10) .15996706E-02 (17,14) .69593818E-04
(17,15)-.69593818E-04 (17,16) .69593818E-04 (17,17) .22302577E+00
(17,18) .72029601E-04 (17,19) .85659253E-01 (17,20) .47832013E+02
(17,21) .74646919E+02 (17,22) .26074716E-05 (17,23) .52149431E-03
(17,25) .11175732E+02 (17,26) .19047635E+00 (17,27)-.18866682E+02
(17,28) .52118142E-03 (18, 3) .50880579E-01 (18, 4) .18970107E-03
(18, 5)-.10483536E-05 (18, 6)-.27490975E-08 (18, 7)-.12674943E-04
(18,11)-.60081997E+00 (18,12) .10540411E+00 (18,13) .99487972E-03
(18,14) .23544716E-07 (18,15)-.23544716E-07 (18,16) .23544716E-07
(18,17)-.40026017E-07 (18,18) .74067936E+00 (18,19)-.62600690E-08
(18,20)-.12520138E-05 (18,22) .55714815E-01 (18,23) .24864939E+02
(18,24) .27791629E+02 (18,25)-.12512626E-05 (18,28)-.86701009E+00
(19, 1)-.22385270E+01 (19, 2)-.96692790E-02 (19, 5) .11175701E-01
(19, 6) .30356597E-04 (19, 7) .13143046E+00 (19, 8) .13789607E+03
(19, 9)-.18636239E+00 (19,10)-.82281809E-02 (19,14)-.26893282E-03
(19,15) .26893282E-03 (19,16)-.26893282E-03 (19,17) .45718579E-03
(19,18)-.27834547E-03 (19,19) .71503858E-04 (19,20)-.42432482E+03
(19,21)-.43748500E+03 (19,22)-.10076106E-04 (19,23)-.20152212E-02
(19,25)-.16805239E+03 (19,26)-.10338663E+01 (19,27) .16838297E+03
(19,28)-.20140120E-02 (20, 1)-.13677512E-04 (20, 2)-.54725198E-07
(20, 5) .61041976E-07 (20, 6) .16048118E-09 (20, 7) .74708801E-06
(20, 8) .86133725E-03 (20, 9)-.33166224E-05 (20,10)-.58330592E-07
(20,14)-.13933326E-08 (20,15) .13933326E-08 (20,16)-.13933326E-08
(20,17) .23686654E-08 (20,18)-.14420993E-08 (20,19) .37045927E-09
(20,20) .16024267E+01 (20,21) .21216956E+01 (20,22)-.52203993E-10
(20,23)-.10440799E-07 (20,25)-.90316616E-03 (20,26)-.74024734E-05
(20,27) .14107523E-02 (20,28)-.10434534E-07 (21, 1) .17946481E-04
(21, 2) .69218069E-07 (21, 5)-.75997308E-07 (21, 6)-.19708420E-09
(21, 7)-.94667766E-06 (21, 8)-.87739231E-03 (21, 9) .69657548E-05
(21,10) .83124675E-07 (21,14) .16969105E-08 (21,15)-.16969105E-08
(21,16) .16969105E-08 (21,17)-.28847479E-08 (21,18) .17563024E-08
(21,19)-.45117457E-09 (21,20)-.21191497E+01 (21,21)-.21848001E+01
(21,22) .63578147E-10 (21,23) .12715629E-07 (21,25) .24972360E-02
(21,26) .10996661E-04 (21,27)-.24631143E-02 (21,28) .12708000E-07
(22, 3)-.74624900E+00 (22, 4)-.27562442E-02 (22, 5) .15183560E-04
(22, 6) .39648848E-07 (22, 7) .18461803E-03 (22,11)-.93497819E+01
(22,12)-.16614717E+01 (22,13)-.14867662E-01 (22,14)-.33874197E-06
(22,15) .33874197E-06 (22,16)-.33874197E-06 (22,17) .57586136E-06
```

```
(22,18)-.35059794E-06 (22,19) .90064716E-07 (22,20) .18012943E-04
(22,22)-.12691646E-07 (22,23)-.42433913E+03 (22,24)-.43748500E+03
(22,25) .18002135E-04 (22,28) .90770122E-02 (23, 3)-.43740972E-05
(23, 4)-.15261326E-07 (23, 5) .82301375E-10 (23, 6) .20844650E-12
(23, 7) .10393623E-08 (23,11)-.36179713E-03 (23,12)-.12586335E-04
(23,13)-.95299123E-07 (23,14)-.17478962E-11 (23,15) .17478962E-11
(23,16)-.17478962E-11 (23,17) .29714235E-11 (23,18)-.18090725E-11
(23,19) .46473063E-12 (23,20) .92946127E-10 (23,22)-.65488426E-13
(23,23) .16024266E+01 (23,24) .21216956E+01 (23,25) .92890359E-10
(23,28) .50291243E-03 (24, 3) .56264496E-05 (24, 4) .19115639E-07
(24, 5)-.10211597E-09 (24, 6)-.25538595E-12 (24, 7)-.13108561E-08
(24,11) .10143474E-02 (24,12) .19146654E-04 (24,13) .12884144E-06
(24,14) .21251103E-11 (24,15)-.21251103E-11 (24,16) .21251103E-11
(24,17)-.36126874E-11 (24,18) .21994891E-11 (24,19)-.56502431E-12
(24,20)-.11300486E-09 (24,22) .79621506E-13 (24,23)-.21191496E+01
(24,24)-.21848001E+01 (24,25)-.11293706E-09 (24,28) .45343284E-04
(25, 1) .18895334E-01 (25, 2) .12403035E-03 (25, 5)-.18014936E-03
(25, 6)-.70402890E-06 (25, 7)-.16180082E-02 (25, 8)-.40429470E+00
(25, 9)-.20421261E-02 (25,10) .37776022E-04 (25,14) .87660992E-05
(25,15)-.87660992E-05 (25,16) .87660992E-05 (25,17)-.14902369E-04
(25,18) .90729127E-05 (25,19)-.23307305E-05 (25,20)-.46614609E-03
(25,22) .32843944E-06 (25,23) .65687888E-04 (25,25) .79502985E+00
(25,26) .65084798E-02 (25,27)-.79432103E+00 (25,28) .65648475E-04
(26, 1)-.30299604E-01 (26, 2)-.19902157E-03 (26, 5) .28916660E-03
(26, 6) .11307518E-05 (26, 7) .25960847E-02 (26, 8) .64127573E+00
(26, 9) .32896449E-02 (26,10)-.60512050E-04 (26,14)-.14087229E-04
(26,15) .14087229E-04 (26,16)-.14087229E-04 (26,17) .23948290E-04
(26,18)-.14580282E-04 (26,19) .37455126E-05 (26,20) .74910251E-03
(26,22)-.52780623E-06 (26,23)-.10556125E-03 (26,25)-.56836776E-02
(26,26) .78502748E+00 (26,27) .12840944E+01 (26,28)-.10549791E-03
(27, 1) .18808325E-01 (27, 2) .12360617E-03 (27, 5)-.17963545E-03
(27, 6)-.70269525E-06 (27, 7)-.16122630E-02 (27, 8)-.40471312E+00
(27, 9)-.20470334E-02 (27,10) .37531620E-04 (27,14) .87568294E-05
(27,15)-.87568294E-05 (27,16) .87568294E-05 (27,17)-.14886610E-04
(27,18) .90633185E-05 (27,19)-.23282658E-05 (27,20)-.46565316E-03
(27,22) .32809213E-06 (27,23) .65618426E-04 (27,25)-.46537377E-03
(27,28) .65579055E-04 (28, 3) .83069000E-02 (28, 4) .43106613E-04
(28, 5)-.25902601E-06 (28, 6)-.97562883E-09 (28, 7)-.23945062E-05
(28,11)-.33828328E+00 (28,12) .76689422E-02 (28,13) .11493097E-03
(28,14) .11771007E-07 (28,15)-.11771007E-07 (28,16) .11771007E-07
(28,17)-.20010712E-07 (28,18) .12182992E-07 (28,19)-.31296753E-08
(28,20)-.62593507E-06 (28,22) .44102432E-09 (28,23) .88204863E-07
(28,25)-.62555950E-06 (28,28) .88151940E-07 (29,14)-.10000000E+01
(29,15) .10000000E+01 (29,16)-.10000000E+01 (29,17) .17000000E+01
(29,18)-.10350000E+01 (29,19) .26588000E+00 (29,20) .53176000E+02
(29,22)-.37467000E-01 (29,23)-.74934000E+01 (29,25) .53144094E+02
(29,28)-.74889040E+01
MATRIX B          IS
```

```
( 1, 1) .20216331E-02 ( 2, 1) .29773005E-02 ( 3, 1) .80949911E-05
( 4, 1)-.12375691E-02 ( 5, 1)-.10778218E+00 ( 6, 1)-.17330942E+02
( 7, 1)-.23937219E-01 ( 8, 1) .27020656E-05 ( 9, 1) .96485786E-03
(10, 1) .35483933E+00 (11, 1) .42431595E-08 (12, 1) .14120225E-05
(13, 1) .39982020E-03 (14, 1) .24990000E-01 (17, 1) .18268377E-04
(18, 1) .61804879E-08 (19, 1)-.70594865E-04 (20, 1)-.36574981E-09
(21, 1) .44543902E-09 (22, 1)-.88919768E-07 (23, 1)-.45882275E-12
(24, 1) .55784144E-12 (25, 1) .23011010E-05 (26, 1)-.36978977E-05
(27, 1) .22986677E-05 (28, 1) .30898893E-08 (29, 1)-.26250000E+00
MATRIX CD          IS
( 1, 1) .10000000E+01 ( 2, 2) .10000000E+01 ( 2, 4) .17000000E+01
( 2, 6) .26588000E+00 ( 2, 7) .53176000E+02 ( 2,12) .53144094E+02
( 3, 3) .10000000E+01 ( 3, 5) .10350000E+01 ( 3, 9) .37467000E-01
( 3,10) .74934000E+01 ( 3,15) .74889040E+01 ( 4, 4) .10000000E+01
( 4, 6) .15640000E+00 ( 4, 7) .31280000E+02 ( 4,12) .31261232E+02
( 5, 5) .10000000E+01 ( 5, 9) .36200000E-01 ( 5,10) .72400000E+01
( 5,15) .72356560E+01 ( 6, 1)-.10000000E+01 ( 6, 2) .10000000E+01
( 6, 3)-.10000000E+01 ( 6, 4) .17000000E+01 ( 6, 5)-.10350000E+01
( 6, 6) .26588000E+00 ( 6, 7) .53176000E+02 ( 6, 9)-.37467000E-01
( 6,10)-.74934000E+01 ( 6,12) .53144094E+02 ( 6,15)-.74889040E+01
( 7, 6) .10000000E+01 ( 7, 7) .20000000E+03 ( 7,12) .19988000E+03
( 8, 7) .10000000E+01 ( 8,12) .99940000E+00 ( 9, 9) .10000000E+01
( 9,10) .20000000E+03 ( 9,15) .19988000E+03 (10,10) .10000000E+01
(10,15) .99940000E+00 (11,12) .10000000E+01 (12,15) .10000000E+01

MATRIX FD          IS
(13, 1) .10000000E+01
MATRIX HD          IS
ALL ELEMENTS ARE ZERO
MATRIX D           IS
( 1, 1) .26250000E+00 ( 6, 1)-.26250000E+00

THE COMMON ROOTS ELIMINATED ARE
NO.        REAL            IMAG.            OMEGA            ZETA

 1     -.31294747       .71315815       .77880078        .40183250
 2     -.31294747      -.71315815       .77880078        .40183250
 3      .94101122       .37660536E-01   .94176453       -.99920011
 4      .94101122      -.37660536E-01   .94176453       -.99920011
 5      .79026159       .38037728E-02   .79027075       -.99998842
 6      .79026159      -.38037728E-02   .79027075       -.99998842
 7     -.29118674       .95415875       .99760144        .29188685
 8     -.29118674      -.95415875       .99760144        .29188685
 9     -.29118674       .95415875       .99760144        .29188685
10     -.29118674      -.95415875       .99760144        .29188685
11      1.0000000      0.
12      1.0000000      0.
13      1.0000000      0.
```

```
 14     0.              0.
 15     0.              0.
 16     0.              0.
 17     0.              0.
 18     0.              0.

                    THE NUMERATOR ROOTS OF ZROOT20  ARE

 NO.        REAL             IMAG.              OMEGA              ZETA

  1     -.41065843        .44636207         .60653066          .67706128
  2     -.41065843       -.44636207         .60653066          .67706128
  3     -5.7589334       0.
  4     -.58234746       0.
  5     -.60763833E-01   0.
  6      .22314408       0.
  7      .97044554       0.
  8      .65670000       0.
  9      .90480000       0.
 10      .74067933       0.

                    LOW ORDER NON-ZERO COEFFICIENT =     .48793794E-05

                    THE DENOMINATOR ROOTS OF ZROOT20  ARE

 NO.        REAL             IMAG.              OMEGA              ZETA

  1     -.41054248        .44564184         .60592225          .67754977
  2     -.41054248       -.44564184         .60592225          .67754977
  3      .11183079        .33415974         .35237602         -.31736209
  4      .11183079       -.33415974         .35237602         -.31736209
  5      .78264773        .46194650         .90880803         -.86118047
  6      .78264773       -.46194650         .90880803         -.86118047
  7      .98209324        .38796381E-01     .98285924         -.99922064
  8      .98209324       -.38796381E-01     .98285924         -.99922064
  9      .10724478       0.
 10      .95123134       0.
 11      .73936134       0.

                    LOW ORDER NON-ZERO COEFFICIENT =     .91292656E-03

DEGREE OF NUMERATOR OF ZPTF20  IS 10    (COEFFICIENTS IN ASCENDING ORDER)
.0000048793793967611 .00005422670196196 -.0004301053238645 .00007394109461729
.001529946348813 .0004949714772774 -.003406972769759 -.002694751973388
.007544221803876 -.002500815466003 -.0006726975577703

DEGREE OF DENOMINATOR OF ZPTF20  IS 11  (COEFFICIENTS IN ASCENDING ORDER)
.0009129265648695 -.01389618083208 .06933246273983 -.221239191441 .4571646325039
```

```
-.6147399572023 .879480873489 -1.882099936496 3.152525897068 -3.06864018641
1.573964481339 -.3327693623257
```

The computed closed loop transfer function is stored in **SPTF**$_{20}$. This transfer function is used in Step 6 to compute the step response by recursive evaluation of a difference equation. The response at every .5 sec is

Time	c(t)
.00000	.00000
.50000	.85369
1.0000	1.2902
1.5000	1.2067
2.0000	.95335
2.5000	.80423
3.0000	.80329
3.5000	.86517
4.0000	.90975
final value	.89135

The low resolution printer plot is not presented. However, the high resolution plot is presented in Appendix H.

The step response computed by the state space method in Step 7 is

Time	c(t)
.00000	.00000
.50000	.853$\underline{74}$
1.0000	1.290$\underline{3}$
1.5000	1.2067
2.0000	.953$\underline{41}$
2.5000	.8042$\underline{7}$
3.0000	.803$\underline{33}$
3.5000	.865$\underline{21}$
4.0000	.909$\underline{80}$

The underscored digits represents the difference between the two methods used to compute the time response.

After the loop is opened in Step 8, the command B2TF is used to compute the open loop transfer function. Since the format of the output is the same as above, only the final result of this open loop transfer function is presented next:

```
THE COMMON ROOTS ELIMINATED ARE

NO.          REAL            IMAG.               OMEGA            ZETA
```

```
  1    -.31294747        .71315815       .77880078        .40183250
  2    -.31294747       -.71315815       .77880078        .40183250
  3    -.29118674        .95415875       .99760144        .29188685
  4    -.29118674       -.95415875       .99760144        .29188685
  5     .79026159        .38037728E-02   .79027075       -.99998842
  6     .79026159       -.38037728E-02   .79027075       -.99998842
  7     .94101122        .37660536E-01   .94176453       -.99920011
  8     .94101122       -.37660536E-01   .94176453       -.99920011
  9     1.0000000       0.
 10     1.0000000       0.
 11     1.0000000       0.
 12    0.               0.
 13    0.               0.
 14    0.               0.
 15    0.               0.

               THE NUMERATOR ROOTS OF ZROOT21  ARE

 NO.        REAL            IMAG.           OMEGA           ZETA

  1    -.39529203        .46638977       .61137158        .64656592
  2    -.39529203       -.46638977       .61137158        .64656592
  3    -.29216671        .95633069       .99996489        .29217697
  4    -.29216671       -.95633069       .99996489        .29217697
  5    -51.617763       0.
  6    -2.5632083       0.
  7    -.19110339       0.
  8    -.17540986E-01   0.
  9    -.14233756E-04   0.
 10     .97044554       0.
 11     .96080000       0.
 12     .74067933       0.
 13     1.0000000       0.

               LOW ORDER NON-ZERO COEFFICIENT =     .27926697E-09

               THE DENOMINATOR ROOTS OF ZROOT21  ARE

 NO.        REAL            IMAG.           OMEGA           ZETA

  1    -.41065843        .44636207       .60653066        .67706128
  2    -.41065843       -.44636207       .60653066        .67706128
  3     .26328657        .28713091       .38956896       -.67584072
  4     .26328657       -.28713091       .38956896       -.67584072
  5    -.29118696        .95415864       .99760140        .29188708
  6    -.29118696       -.95415864       .99760140        .29188708
  7     .97879382        .60296480E-01   .98064928       -.99810793
  8     .97879382       -.60296480E-01   .98064928       -.99810793
 9      .66670000        0.
10      .22314408        0.
11      .52543884        0.
12      .70477617        0.
13      .94753751        0.
14     0.                0.

                    LOW ORDER NON-ZERO COEFFICIENT =    .93256827E-03

DEGREE OF NUMERATOR OF ZPTF21  IS 13    (COEFFICIENTS IN ASCENDING ORDER)
2.792669697423E-10 .00001963706374265 .001195505178997 .004241776865733
-.008278125644938 -.006740054966407 -.01146909059935 .04160133030724
-.01510067489718 .01702852177761 -.03511243507414 .003514175091163
.008928046689292 .0001713879289734

DEGREE OF DENOMINATOR OF ZPTF21  IS 14  (COEFFICIENTS IN ASCENDING ORDER)
0. .0009325682745349 -.01216657050936 .06636119463434 -.2029143353218
.3906411170752 -.5810105254174 1.020725122821 -2.000267252651
2.951993409726 -3.150221313651 2.949423552319 -2.486008740129
1.386835615943 -.3343330135187
```

Note that the order of the closed loop transfer function can be reduced since the zero at -.1423E-4 is nearly zero and its effect is essentially canceled by the pole at the origin. Also the degree of the numerator can be reduced by an additional order since the zero at -51.618 has no effect on the transfer function over the range of frequency of interest. This zero behaves like a zero at minus infinity and thus may be deleted.

In Step 10 the open loop frequency response is evaluated from the rational transfer function computed in Step 9. A portion of the output from this step is given below at 0.01, 0.1, 1.0., and 25. Hz. The low resolution printer plots are not presented. However, the high resolution plots are presented in Appendix H.

```
************************************************************
*     ZFREQ - FREQUENCY RESPONSE OF Z-PLANE TRANSFER        *
*             FUNCTION 21                                   *
************************************************************

SAMPLING PERIOD = .2000E-01

AUTOMATIC FREQUENCY MODE IF FAUTO.NE.0, FAUTO = 1.000

NOMEGA = 4.000    OMEGA = .1000E-01, .1000     , 1.000     , 25.00     ,

 OMEGA     ZREAL ZIMAG    REAL    IMAGINARY      DB       PHASE   PHASE
 HZ                                                                 MARGIN
.1000E-01 1.000  .001   .273E-03  -.898E-02   -40.934   -88.26    91.74
.1300E-01 1.000  .002   .462E-03  -.117E-01   -38.647   -87.74    92.26

                          ⋮

.9600E-01 1.000  .012   .254E-01  -.939E-01   -20.238   -74.87   105.13
.1000     1.000  .013   .276E-01  -.986E-01   -19.798   -74.37   105.63
.1240     1.000  .016   .425E-01  -.129E+00   -17.364   -71.71   108.29

                          ⋮

.9151      .993  .115  -.153E+01   .642E+00     4.409  -202.75   -22.75
1.000      .992  .125  -.138E+01   .768E+00     3.963  -209.13   -29.13
1.160      .989  .145  -.113E+01   .929E+00     3.317  -219.36   -39.36

                          ⋮

24.30     -.996  .088   .677E-02   .126E-02   -43.237  -349.47  -169.47
25.00    -1.000  .000   .683E-02   .217E-15   -43.318  -360.00  -180.00
```

In Step 11 the open loop frequency response is evaluated directly from the state space representation. A portion of the output from this step is given below at 0.01, 0.1, 1.0., and 25. Hz. No plots are presented since the response is essentially identical to that computed in Step 10.

```
                          ⋮

INPUT IS CONNECTED TO BLOCK D6
MAGNITUDE OF INPUT =  1.0000
BLOCK D2   IS THE OUTPUT

**********************************************************************
*    B2FREQ - BLOCK2: FREQUENCY RESPONSE FROM D6   TO D2             *
**********************************************************************

BASIC SAMPLING PERIOD COMPUTED =  .50000E-03
LCM SAMPLING PERIOD COMPUTED =  .20000E-01
TOTAL NUMBER OF TIME SEGMENTS COMPUTED =    40

                          ⋮

SAMPLING PERIOD = .2000E-01
```

```
AUTOMATIC FREQUENCY MODE IF FAUTO.NE.0, FAUTO = 1.000

NOMEGA = 4.000    OMEGA = .1000E-01, .1000     , 1.000     , 25.00     ,

 OMEGA     ZREAL ZIMAG    REAL    IMAGINARY      DB      PHASE    PHASE
  HZ                                                              MARGIN
.1000E-01 1.000  .001   .273E-03  -.898E-02   -40.934   -88.26    91.74
.1300E-01 1.000  .002   .462E-03  -.117E-01   -38.647   -87.74    92.26

                           :

.9600E-01 1.000  .012   .254E-01  -.939E-01   -20.238   -74.87   105.13
.1000     1.000  .013   .276E-01  -.986E-01   -19.799   -74.37   105.63
.1240     1.000  .016   .425E-01  -.129E+00   -17.364   -71.71   108.29

                           :

.9151      .993  .115  -.153E+01   .642E+00     4.409  -202.75   -22.75
1.000      .992  .125  -.138E+01   .768E+00     3.962  -209.13   -29.13
1.160      .989  .145  -.113E+01   .929E+00     3.316  -219.36   -39.36

                           :

24.30     -.996  .088   .677E-02   .126E-02   -43.238  -349.46  -169.46
25.00    -1.000  .000   .682E-02   .248E-09   -43.318  -360.00  -180.00
```

The only difference between the tabular output from these frequency response commands are the last significant digits for dB at omega = 0.1 and 1.0.

The system eigenvalues could have been computed with command B2EIG. However, since the closed loop transfer function was computed in Step 5, the eigenvalues are equal to the denominator roots of $\mathbf{ZPTF}_{20}$ plus the common roots which were eliminated.

9.8 Example 21 IEEE CACSD Benchmark Problem No. 3, 4-Rate Model (classical multirate transform method)

Solution of the 4-rate sampled-data model from the IEEE CACSD Benchmark Problem No. 3 [5] is presented in this example using classical multirate transform methods. This method of analysis is based on the block diagram reduction method in which fast rate transforms are converted to the slowest sampling rate. The block diagram of this problem, labeled in terms of the LCAP2 **SPTF**$_i$ and **ZPTF**$_i$ transfer functions, is given in Figure 9.11.

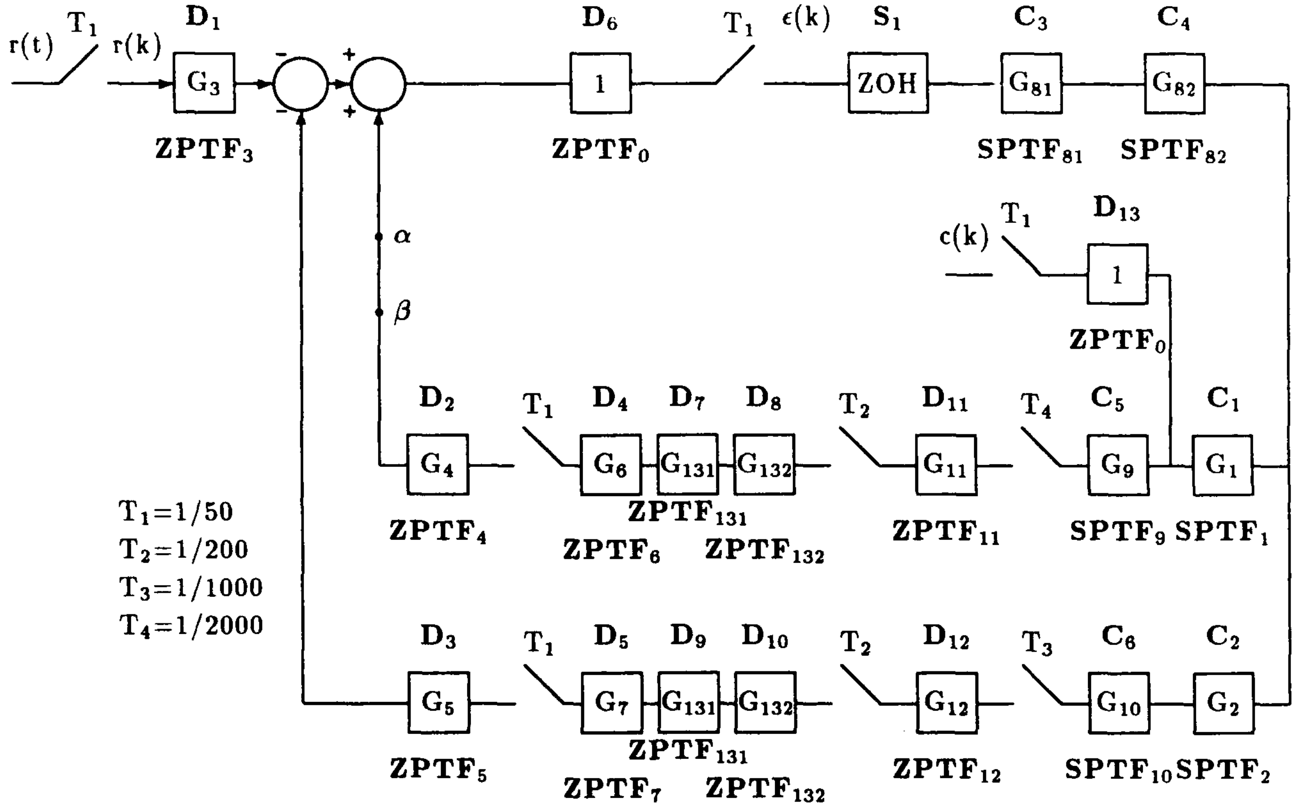


Figure 9.11: Block Diagram for IEEE Benchmark 4-Rate Model

Problem statement, system parameters, and transfer functions are the same as those given in Example 20.

The same methodology as in Example 19 will be used to reduce the block diagram to a single rate rate system. For computational reasons, however, the analysis will be done in the w plane instead of the z plane. A change in notation for defining transforms with different sampling periods will be made so that it will be more explicit and also consistent with that used by LCAP2 commands. The sampling period of the slowest sampler will be denoted by T so that the w transform of a continuous transform G(s) is denoted by

$$\mathcal{W}^T\{G(s)\} = G^T(w)$$

If G(s) is sampled at a rate n (an integer) times faster, the faster rate w transform of G(s) is denoted by

$$\mathcal{W}^{T/n}\{G(s)\} = G^{T/n}(w_n)$$

where $w_n = (z_n-1)/(z_n+1)$ and $z_n = [z]^{1/n}$ since $z = e^{sT}$ and $z_n = e^{sT/n}$.

The first step in simplifying the block diagram is to compute the w transforms from $\epsilon^T(w)$ to the outputs of $G_9(s)$ and $G_{10}(s)$ at the sampling rates corresponding to periods $T_4 = T/40$ and $T_3 = T/20$, respectively. Before doing this though, the following block diagram reduction is performed

$$G_{19} = G_9 * G_1 * G_{81} * G_{82}$$

and

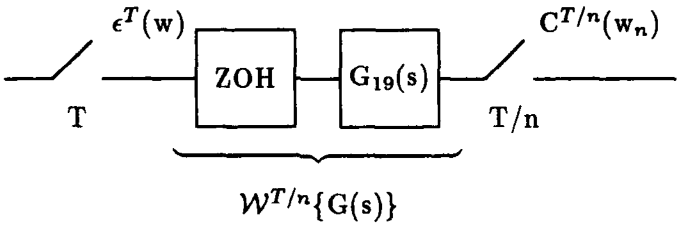
$$G_{21} = G_{10} * G_2 * G_{81} * G_{82}$$

so that there will be only a single s plane transfer function between the ZOH and the output of the two fastest samplers.

Applying the slow-to-fast multirate relationship

$$C^{T/n}(w_n) = \mathcal{W}^{T/n}\{G(s)\} * E^T(w)$$

to the following system



the transfer function between $\epsilon^T(w)$ and the output of $G_{19}(s)$ at the sampling rate corresponding to period T/40 is given by[1]

$$G_{34}^{T/40}(w_{40}) = \text{w transform of } \mathcal{Z}^{T/40}\{\frac{1 - z_{40}^{-40}}{s}\ G_{19}(s)\}$$

which is to be computed with command SWMRX. The arguments of SWMRX will be selected so that the resulting w transform will be stored in $\mathbf{WPTF}_{34}$. Similarly, the transfer function from $\epsilon^T(w)$ to the output of $G_{21}(s)$ at the sampling rate corresponding to period T/20 will be computed and stored in $\mathbf{WPTF}_{50}$.

Next, transform each of the z plane transfer functions into the w plane and store the results into the $\mathbf{WPTF}_i$'s with the same identifiers, i.e., $\mathbf{WPTF}_i$ = w transform of $\mathbf{ZPTF}_i$. Computing the products

[1] See Example 19 for s plane equivalent

$$\mathbf{WPTF}_{61} = \mathbf{WPTF}_5 * \mathbf{WPTF}_{131} * \mathbf{WPTF}_{132}$$

and

$$\mathbf{WPTF}_{72} = \mathbf{WPTF}_7 * \mathbf{WPTF}_{131} * \mathbf{WPTF}_{132}$$

Figure 9.11 can now be redrawn in terms of w plane transforms as shown in Figure 9.12.

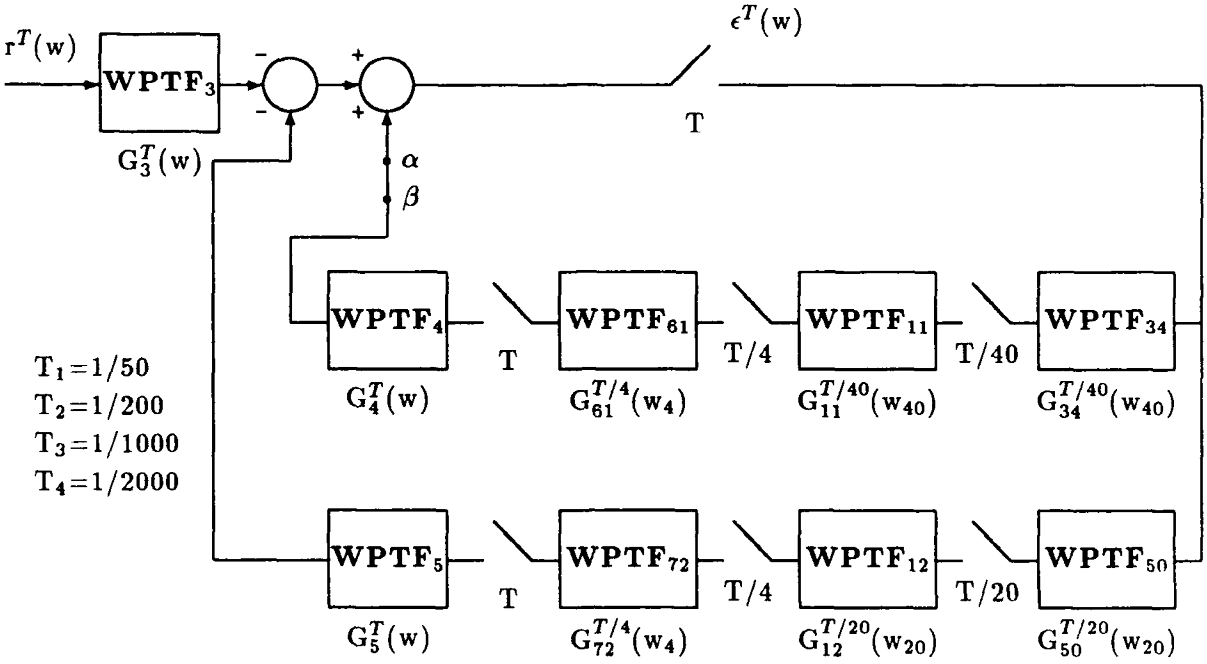


Figure 9.12: Multirate Rate W Plane Transfer Function Representation for 4-Rate Model

The next step in simplifying the block diagram is to transform the faster rate w transforms to a slower rate. First compute the products

$$G_{40}^{T/40}(w_{40}) = G_{11}^{T/40}(w_{40}) * G_{34}^{T/40}(w_{40})$$

and

$$G_{51}^{T/20}(w_{20}) = G_{12}^{T/20}(w_{20}) * G_{50}^{T/20}(w_4)$$

and store them into **WPTF**$_{40}$ and **WPTF**$_{51}$, respectively. Then using the w plane equivalent of the z plane fast-to-slow multirate relationship

$$C^T(z) = \frac{1}{n}\sum_{k=0}^{n-1} C^{T/n}(z_n e^{\frac{j2\pi*k}{n}})$$

the transfer function between $\epsilon^T(w)$ and the sampled output[1] of $G_{40}{}^{T/40}(w_{40})$ is computed with command WMRXFM. The arguments of WMRXFM will be selected so that the resultant transform

[1] Sampling period = T/4

is stored in **WPTF$_{41}$**. Similarly, the transfer function between ϵ^T(w) and the sampled output[1] of $G_{51}{}^{T/20}(w_{20})$ is computed and stored in **WPTF$_{52}$**.

Applying this same procedure to the transfer functions,

$$G_{42}^{T/4}(w_{T/4}) = G_{61}^{T/4}(w_4) * G_{41}^{T/4}(w_4)$$

and

$$G_{53}^{T/4}(w_4) = G_{72}^{T/4}(w_4) * G_{52}^{T/4}(w_4)$$

compute transfer functions from ϵ^T(w) to the sampled outputs[2] of $G_{42}{}^{T/4}(w_4)$ and $G_{53}{}^{T/4}(w_4)$, and then store into **WPTF$_{43}$** and **WPTF$_{54}$**, respectively.

The block diagram can now be simplified to the single rate system in Figure 9.13.

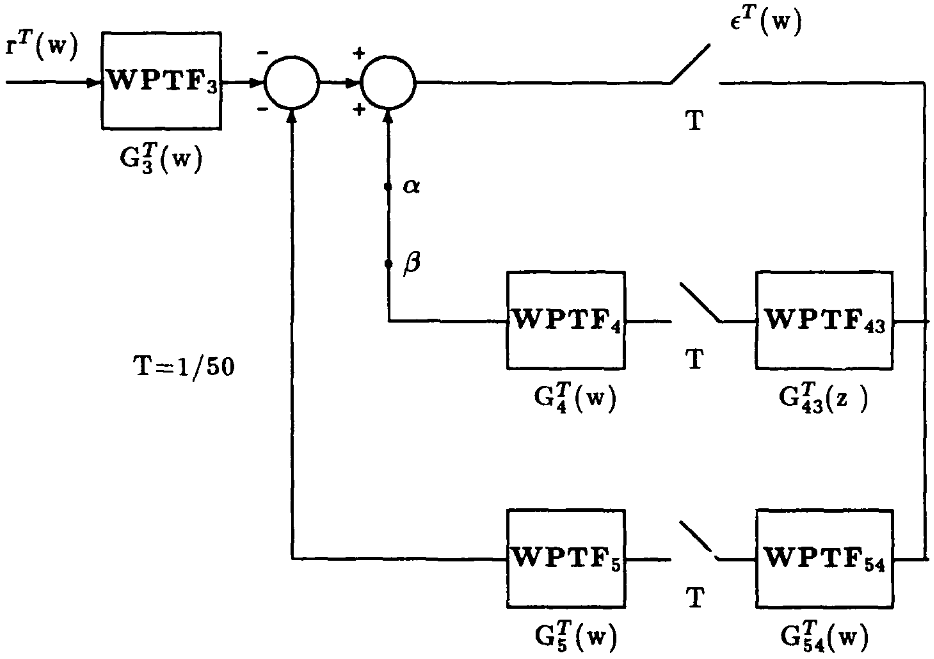


Figure 9.13: Slow Rate W Plane Transfer Function Representation for 4-Rate Model

By inspection from Figure 9.13, the open loop transfer function with the inner loop broken between α and β, as defined in [5], is

$$\frac{G_4^T(w) * G_{43}^T(w)}{1 + G_5^T(w) * G_{54}^T(w)}$$

[1] Sampling period = T/4

[2] Sampling period = T

The closed loop transfer function from $r^T(w)$ to the output of c(s) sampled at the slowest sampling rate is not readily determined from Figure 9.13 since the output does not appear explicitly. The desired closed loop transfer function, though, can be computed as

$$\frac{c^T(w)}{R^T(w)} = \frac{\epsilon^T(w)}{R^T(w)} * \mathcal{W}^T\{\frac{-2w}{(1-w)s}G_1(s) * G_{81}(s) * G_{82}(s)\}$$

where

$$\frac{\epsilon^T(w)}{R_T(w)} = \frac{-G_3^T(w)}{1 + G_5^T(w) * G_{54}^T(w) - G_4^T(w) * G_{43}^T(w)}$$

Since there are many steps involved in the solution of this problem, the PRECMP precompiler will be used. The PRECMP code is:

```
C     IEEE CACSD BENCHMARK PROBLEM NO. 3, 4-RATE MODEL
C
C     1.  **** LOAD IN S PLANE TRANSFER FUNCTIONS ****
*POLYN 1 -15*1.5 -15
*POLYD 2 13 6 1
*SPLDC 1
C
*POLYN 1 9 -.1
*POLYD 2 13 6 1
*SPLDC 2
C
*POLYN 0 200.*200.
*POLYD 3 0. 200.*200. 2.*.125*200. 1.
*SPLDC 9
C
*POLYN 0 100.*100.
*POLYD 3 0. 100.*100. 2.*.125*100. 1.
*SPLDC 10
C
*POLYN 0 62.8*62.8
*POLYD 2 62.8*62.8 94.25 1
*SPLDC 81
C
*POLYN 0 1
*POLYD 1 1 .03
*SPLDC 82
C
C     2.  **** ENTER SAMPLING PERIODS ****
      T1=1/50.
      T2=1/200.
      T3=1/1000.
      T4=1/2000.
C
C     3.  **** LOAD IN Z PLANE TRANSFER FUNCTIONS ****
*POLYN 1 -.2625*.9048 .2625
*POLYD 1 -1 1
*ZPLDC 3
C
*POLYN 1 -1.7*.9608 1.7
*POLYD 1 -.6667 1
*ZPLDC 4
C
*POLYN 1 -1.035*.9324 1.035
*POLYD 1 -1 1
*ZPLDC 5
C
*POLYN 1 .1564 .1564
*POLYD 1 -.6873 1.0
*ZPLDC 6
C
*POLYN 1 .0362 .0362
*POLYD 1 -.9277 1
*ZPLDC 7
C
*POLYN 2 .9942 -.9942*1.6179 .9942
*POLYD 3 0. .9883 -1.6084 1.
*ZPLDC 11
C
*POLYN 0 1
*POLYD 1 0 1
*ZPLDC 12
C
*POLYN 0 1
*POLYD 0 1
*ZPLDC 15
C
*POLYN 1 -1 1
*POLYD 1 0 T2
*ZPLDC 131
C
*POLYN 2 .9994 -.9994*1.786 .9994
*POLYD 2 .9988 -1.785 1.
*ZPLDC 132
C
C     4.  ***** COMPUTE S-W TRANSFORM OF INNER LOOP *****
C
*SPMPY 8 81 82
```

```
*SPRTS 8       ! ALSO USED LATER IN STEP 5
*SPRTS 9
*SPRTS 1
*SPMPY 19 1 8
*SPEQU 120 19  ! SAVE FOR LATER USE IN STEP 19
*SPMPY 19 19 9
      SAMPT=T4
      NTGER=40
*SWMRX 34 19 ! WPTF34 IS THE TRANSFORM FROM ETA TO THE SAMPLED OUTPUT
C              OF G9(S) AT T4
C
C     5.  ***** COMPUTE S-W TRANSFORM OF OUTER LOOP *****
C
*SPRTS 2
*SPRTS 10
*SPMPY 20 8 2
*SPMPY 21 20 10
      SAMPT=T3
      NTGER=20
*SWMRX 50 21 ! WPTF50 IS THE TRANSFER FUNCTION FROM ETA TO THE SAMPLED
C              OUTPUT G10(S) WITH SAMPLING PERIOD T3
C
C     6.  ***** TRANSFORM ALL THE Z PLANE TRANSFER FUNCTIONS TO THE W
C         ***** PLANE SO THAT BLOCK DIAGRAM REDUCTION CAN BE CARRIED OUT
C         ***** WITH BETTER ACCURACY LATER
C
      SAMPT=T1
*ZPRTS 3
*ZWXFM 3 3
*ZWXFM 4 4
*ZWXFM 5 5
      SAMPT=T2
*ZWXFM 6 6
*ZWXFM 7 7
*ZWXFM 131 131
*ZWXFM 132 132
      SAMPT=T3
*ZWXFM 12 12
      SAMPT=T4
*ZWXFM 11 11
C
C     7.  ***** MULTIPLY G5*G131*G132 AND STORE INTO G61
C         ***** MULTIPLY G7*G131*G132 AND STORE INTO G72
C
*WPMPY 60 6 131
*WPMPY 61 60 132
C
*WPMPY 71 7 131
```

```
*WPMPY 72 71 132
C
C     8.  ***** MULTIPLY G11 AND G34 AND TRANSFORM DOWN TO T2
C
*WPMPY 40 11 34
      SAMPT=T2
      NTGER=10
*WMRXFM 41 40 ! WPTF41 IS THE FAST-TO-SLOW TRANSFORM FROM ETA TO
C               THE SAMPLED OUTPUT OF G11
C
C     9.  ***** MULTIPLY G12 AND G50 AND TRANSFORM DOWN TO T2
*WPMPY 51 12 50
      SAMPT=T2
      NTGER=5
*WMRXFM 52 51 ! WPTF52 IS THE FAST-TO-SLOW TRANSFORM FROM ETA TO
C               THE SAMPLED OUTPUT OF G50
C
C     10.  ***** MULTIPLY G61 AND G41 AND TRANSFORM DOWN TO T1
C
*WPMPY 42 61 41
*WELCR 42      ! ELIMINATE COMMON ROOTS
      SAMPT=.02
      NTGER=4
*WMRXFM 43 42 ! WPTF43 IS THE TRANSFORM FROM ETA TO THE OUTPUT OF G6
C
C     11.  ***** MULTIPLY G72 AND G52 AND TRANSFORM DOWN TO T1
C
*WPMPY 53 72 52
*WELCR 53
      SAMPT=.02
      NTGER=4
*WMRXFM 54 53 ! WPTF54 IS THE TRANSFORM FROM ETA TO THE OUTPUT OF G7
C
C     12.  ***** COMPUTE INNER LOOP TRANSFER FUNCTION
C
*WPMPY 44 4 43 ! WPTF44 IS INNER LOOP TRANSFER FUNCTION AT THE SLOWEST
C                RATE
*WZXFM 44 44   ! COMPUTE EQUIVALENT Z PLANE TRANSFER FUNCTION
C
C     13.  ***** COMPUTE OUTER LOOP TRANSFER FUNCTION
C
*WPMPY 55 5 54 ! WPTF55 IS THE OUTER LOOP TRANSFER FUNCTION AT THE
C                SLOWEST RATE
*WZXFM 55 55
C
C     14.  ***** COMPUTE OPEN INNER LOOP, CLOSED OUTER LOOP TRANSFER
C                FUNCTION
C
```

```
*CPYWP 55 6 7
*PADD 8 7 6
*CPYPW 60 7 8 ! WPTF60 IS THE CLOSED OUTER LOOP ONLY T.F.
*WELCR 60
*WPMPY 61 60 44 ! WPTF61 IS THE CLOSED LOOP T.F. IN SERIES WITH OPEN
C                  INNER LOOP, I.E., BENCHMARK OPEN LOOP T.F.
*WELCR 61
*WZXFM 61 61  ! COMPUTE EQUIVALENT Z PLANE TRANSFER FUNCTION
C
C     15.  ***** COMPUTE OPEN LOOP TRANSFER FUNCTION
C
      TITLE2='OPEN INNER LOOP, CLOSED OUTER LOOP'
      NOMEGA=4
*OMEGA .01 .1 1. 25
      RAD=0
      CALL ZFREQ(61)
C
C     16. ***** COMPUTE CLOSED LOOP TRANSFER FUNCTION FROM R TO ETA
C               COMPUTE G55-G44 FIRST, THEN ADD 1
*WPSUB 45 55 44
*WPADD 46 0 45   ! FORM CLOSED LOOP DENOMINATOR
*CPYWP 46 1 2
*CPYPW 47 2 1
*ROOTN -1
*ROOTD 1
*WPLDR 35  ! LOAD IN NEGATIVE UNITY TRANSFER FUNCTION
*WPMPY 36 35 3  ! NEGATIVE OF G3
*WPMPY 48 36 47  ! R TO ETA TRANSFER FUNCTION
*WZXFM 48 48
C
C     17. ***** COMPUTE TRANSFER FUNCTION FROM ETA TO OUTPUT OF G1
C
      SAMPT=T1
*SWXFM 73 120  ! SPTF120 =G1*G81*G82 FROM STEP 4
*WZXFM 73 73   ! Z PLANE EQUIVALENT
C
C     18. ***** COMPUTE CLOSED LOOP T.F. FROM R TO C
C
*WPMPY 74 48 73
*WELCR 74
*WZXFM 74 74
C
C     19. ***** COMPUTE TIME RESPONSE
C
      TEND=4
*ZTIME 74
```

Since the complete output of the example is too long to be included in this report, only a portion of it will be presented. The output from the last part of Step 14 for the open loop transfer function is:

```
                    THE NUMERATOR ROOTS OF WROOT61  ARE

NO.       REAL            IMAG.              OMEGA            ZETA

 1     -1.0733873       1.5999544         1.9266589       0.55712367
 2     -1.0733873      -1.5999544         1.9266589       0.55712367
 3     0.00000000       1.3511342         1.3511342       0.00000000
 4     0.00000000      -1.3511342         1.3511342       0.00000000
 5    -0.14897670      0.00000000
 6    -0.19991840E-01  0.00000000
 7     -1.0357747      0.00000000
 8     -1.4737498      0.00000000
 9    -0.14998874E-01  0.00000000
10      2.2700451      0.00000000
11      1.0000000      0.00000000
12      1.0000000      0.00000000
13     0.00000000      0.00000000

                   LOW ORDER NON-ZERO COEFFICIENT =  -0.13312953E+20

                    THE DENOMINATOR ROOTS OF WROOT61  ARE

NO.       REAL            IMAG.              OMEGA            ZETA

 1    -0.97634013E-02  0.30764673E-01   0.32276758E-01   0.30249015
 2    -0.97634013E-02 -0.30764673E-01   0.32276758E-01   0.30249015
 3    -0.50540411      0.34215228       0.61032901       0.82808470
 4    -0.50540411     -0.34215228       0.61032901       0.82808470
 5    -0.33913135E-02  -1.3507008        1.3507050       0.25107729E-02
 6    -0.33913135E-02   1.3507008        1.3507050       0.25107729E-02
 7     -1.1565382      -1.6333429        2.0013470       0.57787992
 8     -1.1565382       1.6333429        2.0013470       0.57787992
 9    -0.31101028      0.00000000
10    -0.26944074E-01  0.00000000
11    -0.17328293      0.00000000
12    -0.19997600      0.00000000
13    -0.63513034      0.00000000

                    LOW ORDER NON-ZERO COEFFICIENT =   0.93280167E+18

DEGREE OF NUMERATOR OF WPTF61  IS 13    (COEFFICIENTS IN ASCENDING ORDER)
0.,  -1.3312952872023E+19,  -1.6399740836515E+21,  -5.4452509355898E+22,
-2.8421610640591E+23,  1.2662810225986E+23,  3.4134558629905E+23,
5.0287701238749E+22,  4.1786774841579E+22,  -1.0373604396277E+23,
-1.12889794467E+23,  -2.3028955865751E+22,  7.236988737753E+21,
1.2691543716862E+22

DEGREE OF DENOMINATOR OF WPTF61  IS 13  (COEFFICIENTS IN ASCENDING ORDER)
9.3280167336848E+17,  6.969290877352E+19,  2.6590748134695E+21,
7.0261008123621E+22,  8.7589499053923E+23,  5.6428241409147E+24,
2.0595550948577E+25,  4.549890087997E+25,  6.4451034098508E+25,
6.2148229594923E+25,  4.4607276009174E+25,  2.4542124379376E+25,
8.3764301370234E+24,  1.7835330185289E+24

*************************************************************
*      ZROOT61  = BILINEAR TRANSFORM OF WROOT61             *
*       ZPTF61  = BILINEAR TRANSFORM OF WPTF61              *
*************************************************************

                    THE NUMERATOR ROOTS OF ZROOT61  ARE

NO.         REAL              IMAG.              OMEGA             ZETA

 1      -0.39540718       0.46654138        0.61156169        0.64655322
 2      -0.39540718      -0.46654138        0.61156169        0.64655322
 3      -0.29217661       0.95636438         1.0000000        0.29217661
 4      -0.29217661      -0.95636438         1.0000000        0.29217661
 5       0.74067933       0.00000000
 6       0.96080000       0.00000000
 7      -0.17573004E-01   0.00000000
 8      -0.19151080       0.00000000
 9       0.97044554       0.00000000
10       -2.5747472       0.00000000
11        1.0000000       0.00000000

                    LOW ORDER NON-ZERO COEFFICIENT =    0.41882625E+22

                    THE DENOMINATOR ROOTS OF ZROOT61  ARE

NO.         REAL              IMAG.              OMEGA             ZETA

 1       0.97882516       0.60289280E-01    0.98068011       -0.99810850
 2       0.97882516      -0.60289280E-01    0.98068011       -0.99810850
 3       0.26328869       0.28712364        0.38956504       -0.67585298
 4       0.26328869      -0.28712364        0.38956504       -0.67585298
 5      -0.29118674      -0.95415875        0.99760144        0.29188685
 6      -0.29118674       0.95415875        0.99760144        0.29188685
 7      -0.41065841      -0.44636208        0.60653066        0.67706126
 8      -0.41065841       0.44636208        0.60653066        0.67706126
 9       0.52554105       0.00000000
```

```
10      0.94752572         0.00000000
11      0.70461868         0.00000000
12      0.66670000         0.00000000
13      0.22314408         0.00000000

                    LOW ORDER NON-ZERO COEFFICIENT =   0.19427413E+24

DEGREE OF NUMERATOR OF ZPTF61  IS 11    (COEFFICIENTS IN ASCENDING ORDER)
4.1882625390881E+21,  2.5461667592719E+23,  8.9565372505345E+23,
-1.7798770917227E+24,  -1.3951755142514E+24,  -2.4039478354571E+24,
8.8891297559332E+24,  -3.3955475726709E+24,  3.6908799257938E+24,
-7.5426920837511E+24,  9.1146461099771E+23,  1.8713071416086E+24

DEGREE OF DENOMINATOR OF ZPTF61  IS 13  (COEFFICIENTS IN ASCENDING ORDER)
1.9427413082725E+23,  -2.5345588105645E+24,  1.3824479248473E+25,
-4.227149613284E+25,  8.1378970293103E+25,  -1.210369810369E+26,
2.1263936207461E+26,  -4.1669902568819E+26,  6.1496375814687E+26,
-6.5625919155831E+26,  6.144285549972E+26,  -5.178888157091E+26,
2.8890745689353E+26,  -6.9648697226538E+25
```

This open loop transfer function is almost identical to the one computed in Example 20 ($\mathbf{ZPTF}_{21}$) using the Kalman-Bertram method. The order of the transfer function is lower since (1) there is not a zero-pole combination near the origin which can be canceled and (2) the is no neglibile zero at -51.617 as in $\mathbf{ZPTF}_{21}$ in Example 20. The differences in the remaining roots between these two examples are underscored above. The coefficients are not the same since they are normalized differently.

A portion of the frequency response from Step 15, together with the difference between Examples 20 and 21 underscored, is given below.

```
 *************************************************************
 *     ZFREQ - FREQUENCY RESPONSE OF Z-PLANE TRANSFER        *
 *             FUNCTION 61                                   *
 *************************************************************

SAMPLING PERIOD =0.2000E-01

AUTOMATIC FREQUENCY MODE IF FAUTO.NE.O, FAUTO = 1.000

NOMEGA = 4.000    OMEGA =0.1000E-01,0.1000    , 1.000    , 25.00    ,

 OMEGA     ZREAL ZIMAG    REAL     IMAGINARY      DB       PHASE   PHASE
  HZ                                                               MARGIN
.1000E-01 1.000  .001   .273E-03  -.898E-02   -40.935   -88.26    91.74
.1300E-01 1.000  .002   .462E-03  -.117E-01   -38.647   -87.74    92.26
```

```
                                    ⋮

.9600E-01 1.000  .012   .254E-01  -.939E-01    -20.238    -74.87    105.13
.1000     1.000  .013   .276E-01  -.986E-01    -19.799    -74.38    105.62
.1240     1.000  .016   .425E-01  -.129E+00    -17.363    -71.72    108.28

                                    ⋮

1.000      .992  .125  -.138E+01   .768E+00      3.961  -209.30    -29.30
1.160      .989  .145  -.113E+01   .933E+00      3.316  -219.55    -39.55

                                    ⋮

25.00    -1.000  .000   .712E-02   .357E-16    -43.955  -360.00   -180.00
```

The output from the last part of Step 18 for the closed loop transfer function is:

```
                THE NUMERATOR ROOTS OF WROOT74  ARE

NO.        REAL            IMAG.            OMEGA           ZETA

 1     -1.1565382      -1.6333429       2.0013470       0.57787992
 2     -1.1565382       1.6333429       2.0013470       0.57787992
 3    -0.49979000E-01  0.00000000
 4    -0.14897670      0.00000000
 5    -0.19997600      0.00000000
 6    -0.63513034      0.00000000
 7      1.0000000      0.00000000
 8      1.4202622      0.00000000
 9    -0.14998872E-01  0.00000000
10     -1.1293899      0.00000000
11     -3.7886695      0.00000000

                LOW ORDER NON-ZERO COEFFICIENT =  -0.40544268E+09

                THE DENOMINATOR ROOTS OF WROOT74  ARE

NO.        REAL            IMAG.            OMEGA           ZETA

 1    -0.86445207E-02  -0.19740489E-01  0.21550282E-01  0.40113260
 2    -0.86445207E-02   0.19740489E-01  0.21550282E-01  0.40113260
 3    -0.65043354      0.49797145       0.81916992      0.79401541
```

```
  4    -0.65043354      -0.49797145       0.81916992       0.79401541
  5    -0.50483130E-01  -0.27183963       0.27648749       0.18258739
  6    -0.50483130E-01   0.27183963       0.27648749       0.18258739
  7     -1.1590403        1.6321899        2.0018537       0.57898351
  8     -1.1590403       -1.6321899        2.0018537       0.57898351
  9    -0.14984642       0.00000000
 10    -0.80745084       0.00000000
 11    -0.24997135E-01   0.00000000

                    LOW ORDER NON-ZERO COEFFICIENT =  -0.45483391E+09

DEGREE OF NUMERATOR OF WPTF74  IS 11    (COEFFICIENTS IN ASCENDING ORDER)
-405442675.4115,  -40540373759.87,  -1029399615415.,  -9014932725520.,
-0.2905634043931E+14,  -0.1908394131918E+14,  0.3478524652566E+14,
0.3917269164053E+14,  2824065789320.,  -0.1050080073541E+14,
-6881340070388.,  -1174303232932.

DEGREE OF DENOMINATOR OF WPTF74  IS 11  (COEFFICIENTS IN ASCENDING ORDER)
-454833912.825,  -40472005447.76,  -2095819469977.,  -0.607804702868E+14,
-5.642996069054ZE+14,  -2.6543265628406E+15,  -9.5569949674168E+15,
-2.0671074218988E+16,  -2.6092151074662E+16,  -1.8928238090245E+16,
-7.4340648839385E+15,  -1.5751814997412E+15

 ************************************************************
 *     ZROOT74  = BILINEAR TRANSFORM OF WROOT74               *
 *      ZPTF74  = BILINEAR TRANSFORM OF WPTF74                *
 ************************************************************

                    THE NUMERATOR ROOTS OF ZROOT74  ARE

NO.        REAL              IMAG.             OMEGA            ZETA

  1    -0.41065841      -0.44636208       0.60653066       0.67706126
  2    -0.41065841       0.44636208       0.60653066       0.67706126
  3     0.90480000       0.00000000
  4     0.74067933       0.00000000
  5     0.66670000       0.00000000
  6     0.22314408       0.00000000
  7     -5.7589334       0.00000000
  8     0.97044554       0.00000000
  9    -0.60763832E-01   0.00000000
 10    -0.58234746       0.00000000

                    LOW ORDER NON-ZERO COEFFICIENT =   0.64182675E+12

                    THE DENOMINATOR ROOTS OF ZROOT74  ARE
NO.        REAL              IMAG.             OMEGA            ZETA
```

```
 1      0.98209992     -0.38792282E-01  0.98286575       -0.99922081
 2      0.98209992      0.38792282E-01  0.98286575       -0.99922081
 3      0.11069011      0.33511920      0.35292659       -0.31363493
 4      0.11069011     -0.33511920      0.35292659       -0.31363493
 5      0.78439407     -0.46175803      0.91021675       -0.86176624
 6      0.78439407      0.46175803      0.91021675       -0.86176624
 7     -0.41054109      0.44561878      0.60590435        0.67756749
 8     -0.41054109     -0.44561878      0.60590435        0.67756749
 9      0.73936272      0.00000000
10      0.10653079      0.00000000
11      0.95122497      0.00000000

                     LOW ORDER NON-ZERO COEFFICIENT =   0.12001725E+15

DEGREE OF NUMERATOR OF ZPTF74  IS 10    (COEFFICIENTS IN ASCENDING ORDER)
641826746728.,  7132904594001.,  -0.5657545358892E+14,  9726111302955.,
2.0124699748364E+14,  0.6510785645237E+14,  -4.4814841451781E+14,
-3.5446394168299E+14,  9.923564104109E+14,  -3.2895377257116E+14,
-0.8848569792934E+14

DEGREE OF DENOMINATOR OF ZPTF74  IS 11  (COEFFICIENTS IN ASCENDING ORDER)
1.200172534403E+14,  -1.831266443757E+15,  9.113889688568E+15,
-2.905814310215E+16,  6.010675784938E+16,  -8.0875336872616E+16,
1.1557351641473E+17,  -2.4731700097239E+17,  4.1453048760663E+17,
-4.0364178929084E+17,  2.0704802617974E+17,  -4.3769624060665E+16
```

This closed loop transfer function is almost identical to the one computed in Example 20 (**ZPTF$_{20}$**) using the Kalman-Bertram method. The underscored digits above represents the difference between these two examples. The coefficients are not the same since they are normalized differently.

It then follows that the time response computed in Step 18 will be very close to the results of Example 20. The output at every 0.5 seconds, with the differences between these two examples underscored, is given below.

Time	c(t)
.00000	.00000
.50000	.85386
1.0000	1.2904
1.5000	1.2068
2.0000	.95341
2.5000	.80422
3.0000	.80327
3.5000	.86519
4.0000	.90982
final value	.89141

While the differences between the open and closed loop results with Example 20 are too small to be noticed in a plot, the poles and zeros of both the open and closed loop transfer functions are sufficiently different to warrant a discussion. Relative to Example 20[1], note that the open loop poles are more accurate than the open loop zeros while the opposite it true with the closed loop poles and zeros. This will be explained below.

The single largest contributor to the difference between the results of Examples 20 and 21 is most likely the accuracy in computing the numerator roots of the fast-to-slow rate transform from T/40 to T/4 in Step 8. This operation transforms $\mathbf{WPTF}_{40}$, which is a 49th/50th order transfer function with 38 poles at w=-1.0, into $\mathbf{WPTF}_{41}$, which is a 10th/11th order transfer function. As discussed in Section C.6, the accuracy in computing the numerator roots of a fast-to-slow rate transform can be questionable if the order of the transfer function ($\mathbf{WPTF}_{40}$ in this case) is large, which this example certainly is. To check the accuracy of this transform, command WMRFQ[2], was used in a separate job to compute the multirate frequency response of $\mathbf{WPTF}_{40}$ for comparison with the frequency response of $\mathbf{WPTF}_{41}$. Results of this comparison show that at low frequencies the gain (in dB) and phase agree to at least 3 or 4 significant digits but drop off to 2 or 3 significant digits at higher frequencies. Since this is in the range of the differences found between the open loop frequency responses of Examples 20 and 21, it supports the contention that these differences are most likely attributable to the the accuracy with which the numerator roots of $\mathbf{WPTF}_{41}$ can be calculated.

In computing the open loop transfer function using block diagram reduction, errors associated with this method of analysis can be significant if the order of the transfer functions are too high and if there are too many operations. In this example the order of the transfer functions are not high after transforming down to the slowest sampling rate, but the number of operations are certainly high enough that some degradation in accuracy can be expected. In this example the poles are accurate to at least 5 significant digits but some of the zeros are only accurate to 2 or 3 significant digits. As explained in the preceeding paragraph, the accuracy in these zeroes were not the result of block diagram algebra, but the results of computing a fast-to-slow rate transform for a high order transfer function.

In computing the closed loop transfer function using block diagram reduction, the accuracy of the resultant closed loop poles will have accuracies no better that the open loop zeros since the denominator is 1 plus the open loop transfer function. This is certainly the case since the some of the poles of $\mathbf{ZPTF}_{74}$ are accurate only to 2 or 3 significant digits. The closed loop zeros of $\mathbf{ZPTF}_{74}$, however, are accurate to at least 7 digits. The reason why the zeros are several orders of magnitude more accurate than the poles is dues to the method of computing the zeros. Unlike the poles which are the end results of many block diagram operations, nearly all of the zeros of the closed loop transfer function are computed directly from a single transform operation[3] in Step 17. The order of this transform is not very high so that the accuracy of 7 or 8 significant digits can be expected.

As can be seen, the classical transform method is much more complex to use than the automated analysis method of Example 20 for the analysis of this benchmark problem. It is also not as accurate because of the fast-to-slow rate transform of a high order transfer function. If the ratios of the

[1] An assumption is made that the results obtained with the Kalman-Bertram method are more accurate than the lassical transform method.

[2] It numerically evaluates the frequency response of a fast-to-slow rate transform by direct evaluation using frequency decomposition. See Section C.6 on how this command can be used to check the accuracy of command WMRXFM.

[3] W transform from eta to the output of c at the slowest sampling period

sampling rates were not as high, the accuracy of this method would probably be closer to that of the Kalman-Bertram method.

Appendix A

Reference for LCAP2 Commands

This Appendix provides alphabetical references of all batch LCAP2 commands.

A summary of the notations used for various data types[1] in LCAP2 is given in Table A.1

Table A.1: Summary of Notations Used

notation	description
SPTF$_i$	S plane transfer functions, i=0,999
ZPTF$_i$	Z plane transfer functions, i=0,999
WPTF$_i$	W plane transfer functions, i=0,999
POLY$_i$	Polynomials, i=0,999
C$_i$	S plane transfer function connection block for modeling either continuous or continuous-discrete system i=1,MDIMYC for modeling a continuous system i=1,NDIMYC for modeling a continuous-discrete system
D$_i$	Z plane transfer function connection block for modeling a continuous-discrete system. i=1,NDIMYD
S$_i$	Sample-hold connection block for modeling a continuous-discrete system. i=1,NDIMXS

MDIMYC=30, NDIMYC=NDIMYD=20, and NDIMXS =5 for normal version of LCAP2 on the CRAY and CDC.

MDIMYC=75, NDIMYC=NDIMYD=50, and NDIMXS=10 for large version of LCAP2 on the CRAY[2].

A summary of the LCAP2 commands by subject is given in the following tables.

[1] They are defined in Chapters 6 and 7.

[2] ACCESSed with ID=MX100.

Table A.2: Transfer Function Algebra Commands

PADD	Polynomial add	
PEQU	Polynomial equal	
PSUB	Polynomial subtract	
PMPY	Polynomial multiply	
PLDC	Polynomial load, coefficient form	
PLDR	Polynomial load, root form	
POLY	Print out polynomial	
PDEL	Delete polynomial	
SPADD	S plane transfer function add	*
SPEQU	S plane equal	*
SPSUB	S plane transfer function subtract	*
SPMPY	S plane transfer function multiply	*
SPDIV	S plane transfer function divide	*
SPCLSLP	S plane closed loop transfer function	*
SELCR	Eliminate common roots of an s plane transfer function	*
SNORM	S plane transfer function normalization	*
SPLDC	S plane load, coefficient form	*
SPLDR	S plane load, root form	*
SPTF	Print out s plane transfer function	*
SPDEL	Delete s plane transfer function	*
CPYPS	Copy 2 polynomials into an s plane transfer function	*
CPYSP	Copy s plane transfer function into 2 polynomials	*

* - Corresponding z and w commands are available by substituting Z or W for the letter S.

Table A.3: Frequency Response Commands

SFREQ	Frequency response of $\mathbf{SPTF}_i$
ZFREQ	Frequency response of $\mathbf{ZPTF}_i$
WFREQ	Frequency response of $\mathbf{WPTF}_i$
B1FREQ	SISO frequency response of a continuous system modeled by a connection of s plane transfer functions
B2FREQ	SISO frequency response of a continuous-discrete system modeled by a connection of s and z plane transfer functions and sample-hold blocks
FREQS	Frequency response of a user supplied s plane function
FREQZ	Frequency response of a user supplied z plane function
FREQW	Frequency response of a user supplied w plane function
ZMRFQ	Frequency response of a fast-to-slow multirate z transform
WMRFQ	Frequency response of a fast-to-slow multirate w transform

Table A.4: Time Response Commands

STIME	Time response of $\mathbf{SPTF}_i$
ZTIME	Time response of $\mathbf{ZPTF}_i$
B1TIME	SISO time response of a continuous system modeled by a connection of s plane transfer functions
B2TIME	SISO time response of a continuous-discrete multirate system modeled by a connection of s and z plane transfer functions and sample-hold blocks

Table A.5: Root Finding Commands

PRTS	Find roots of $\mathbf{POLY}_i$
SPRTS	Find roots of $\mathbf{SPTF}_i$
ZPRTS	Find roots of $\mathbf{ZPTF}_i$
WPRTS	Find roots of $\mathbf{WPTF}_i$

Table A.6: Root Locus Commands

SLOCI	Root locus of $\mathbf{SPTF}_i$
ZLOCI	Root locus of $\mathbf{ZPTF}_i$
WLOCI	Root locus of $\mathbf{WPTF}_i$

Table A.7: Transfer Function Transform Commands

SZXFM	S to z plane transform (classical z transform)
SWXFM	S to w plane transform (classical w transform)
ZWXFM	Z to w plane bilinear transform
WZXFM	W to z plane bilinear transform
ZSXFM	Z to "equivalent" s plane root transform
WSXFM	W to "equivalent" s plane root transform
SZMRX	S to z plane slow-fast multirate transform (ZOH at a slower rate)
SWMRX	S to w plane slow-fast multirate transform (ZOH at a slower rate)
ZMRXFM	Z plane fast-slow multirate transform in rational form
WMRXFM	W plane fast-slow multirate transform in rational form

Table A.8: Commands for Determinant of a Polynomial Matrix

DTERM	Determinant of $\mathbf{M}_j(s)$
DETRM	Determinant of $\mathbf{M}(s)$ (old version)

Table A.9: Commands for Automated Analysis of Continuous Systems

B1INIT	Initialization for connection of $\mathbf{C}_i$ blocks
B1CEQ	Equation definition for $\mathbf{C}_i$ connection block
B1END	Termination for connection of $\mathbf{C}_i$ blocks
B1EIG	Compute eigenvalues of a continuous system modeled as a connection of s plane transfer functions
B1TF	Compute transfer function of a continuous system modeled as a connection of s plane transfer functions
B1LOCI	Compute root locus of a continuous system modeled as a connection of s plane transfer functions
B1FREQ	SISO frequency response of a continuous system modeled by a connection of s plane transfer functions
B1TIME	SISO time response of a continuous system modeled by a connection of s plane transfer functions

Table A.10: Commands for Automated Analysis of Continuous-Discrete Multirate Systems

B2INIT	Initialization for connection of $\mathbf{C}_i$, $\mathbf{D}_i$, $\mathbf{S}_i$ blocks
B2CEQ	Equation definition for $\mathbf{C}_i$ connection block
B2DEQ	Equation definition for $\mathbf{S}_i$ connection block
B2SEQ	Equation definition for $\mathbf{S}_i$ connection block
B2END	Termination for connection of $\mathbf{C}_i$, $\mathbf{D}_i$, $\mathbf{S}_i$ blocks
B2EIG	Compute eigenvalues of a continuous-discrete multirate system modeled as a connection of s and z plane transfer functions and sample-hold blocks
B2TF	Compute transfer function of a continuous-discrete multirate system modeled as a connection of s and z plane transfer functions and sample-hold blocks
B2LOCI	Compute root locus of a continuous-discrete multirate system modeled as a connection of s and z plane transfer functions and sample-hold blocks
B2FREQ	SISO frequency response of a continuous-discrete multirate system modeled by a connection of s and z plane transfer functions and sample-hold blocks
B2TIME	SISO time response of a continuous-discrete multirate system modeled by a connection of s and z plane transfer functions and sample-hold blocks

Table A.11: Restart Commands

SAVE	Save all $\mathbf{SPTF}_i$, $\mathbf{ZPTF}_i$,$\mathbf{WPTF}_i$, $\mathbf{POLY}_i$ data, all M0, M1, M2, M3, M4 matrices, and all B0, B1, B2, B3, B4 vectors currently in use
LOAD	Load data saved by the SAVE command
B1SAVE	Save transfer function connection data for a continuous system
B2SAVE	Save transfer function connection data for a continuous-discrete multirate system
B1LOAD	Reload data saved by the B1SAVE command
B2LOAD	Reload data saved by the B2SAVE command

B1CEQ

PURPOSE: Equation definition for a $\mathbf{C}_i$ continuous block used for automated modeling of a continuous system by connection of transfer function blocks

Subroutine B1CEQ is part of a set of subroutines used for defining the connection of s plane transfer functions for automated transfer function analysis of a continuous system. This subroutine is to be called for each $\mathbf{C}_i$ block defined. Before calling this subroutine, the initialization routine B1INIT must first be called. After the last $\mathbf{C}_i$ block is defined by a call to subroutine B1CEQ, the subroutine B1END must be called to complete the connection procedure.

FORTRAN CALL

```
CALL B1CEQ( label, indx, isptf, iyin, nycin, iycin )
```

where, label = label for block $\mathbf{C}_i$ (.LE.60 characters)
indx = block identifier for block $\mathbf{C}_i$ (.LE.MDIMYC)
isptf = **SPTF** identifier for block $\mathbf{C}_i$ (0-999)
iyin = for future use. Enter a dummy value.
nycin = number of continuous blocks connected as inputs to block $\mathbf{C}_i$
iycin = array containing the identifiers of the continuous blocks connected to $\mathbf{C}_i$. (negative value of sign change)

The dummy array iycin can be any integer array dimensioned at least nycin which the user can declare. For convenience, though, an array IYCIN in common block IYCDS can be used instead. This array is described in the following table.

COMMON/IYCDS/ parameters used		
parameter	preset	description
IYCIN	0	Array containing the identifiers of the continuous blocks connected to $\mathbf{C}_i$ IYCIN(1) = id number of 1st continuous block connected to $\mathbf{C}_i$ IYCIN(2) = id number of 2nd continuous block connected to $\mathbf{C}_i$ ⋮ IYCIN(nycin) = id number of nycin-th continuous block connected to $\mathbf{C}_i$

Example 1: Block $\mathbf{C}_3$ is to be represented by s plane transfer $\mathbf{SPTF}_4$. The are two continuous blocks, $\mathbf{C}_1$ and $\mathbf{C}_5$ connected to the input of block $\mathbf{C}_3$. Block $\mathbf{C}_3$ is to be labeled as 'rate gyro filter'. The FORTRAN code to define block $\mathbf{C}_3$ can be written as:

```
INDX=3
ISPTF=4
NYCIN=2
IYCIN(1)=1
IYCIN(2)=5
CALL B1CEQ('RATE GYRO FILTER',INDX,ISPTF,0,NYCIN,IYCIN)
```

PRECMP DIRECTIVE *B1CEQ label indx isptf iyin nycin

produces the following FORTRAN statements

```
 CALL B1CEQ  (
+'label'
+,indx,isptf,iyin,nycin,IYCIN)
```

Note that IYCIN is not an argument in the *B1CEQ directive.

Example 2: Using Example 1 from above, the following code fragment

```
*IYCIN 1 5
*B1CEQ 'rate gyro filter' 3 4 2 0
```

produces the following FORTRAN statements

```
 IYCIN(1)=1
 IYCIN(2)=5
 CALL B1CEQ(
+'RATE GYRO FILTER'
+,3,4,2,0,IYCIN)
```

METHOD See Chapter 7.

COMMENTS See Example 17 in Chapter 9.

PURPOSE: Compute eigenvalues of a continuous system modeled as a connection of s plane transfer functions
$\mathbf{POLY}_i$ = eigenvalues of system matrix, equivalent to det [sI - A]

The eigenvalues of continuous system modeled as a connection[1] of s plane transfer functions are computed. The printout of the state space matrices computed by this command is specified by the parameter PRNMTRX in common block DBASE described by the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
PRNMTRX	0	.EQ.0 No printout of state space matrices .EQ.1 Printout of computed state space matrices .EQ.2 Above plus intermediate matrices

FORTRAN CALL CALL B1EIG(i)

where, i = index for storing polynomial $\mathbf{POLY}_i$

Example 1: Compute the eigenvalues of a continuous system modeled as a connection of s plane transfer functions and store into $\mathbf{POLY}_2$. The FORTRAN code can be written as:

```
CALL B1EIG(2)
```

PRECMP DIRECTIVE *B1EIG i

produces the following FORTRAN statement

```
CALL B1EIG(i)
```

Example 2:

```
*B1EIG 2
```

produces the following FORTRAN statement

```
CALL B1EIG(2)
```

METHOD See Chapter 7 and Section C.2.

[1] With calls to subroutines B1INIT, B1CEQ, and B1END.

B1END

B1END

PURPOSE: Complete the transfer function connection procedure initiated by calls to subroutines B1INIT and B1CEQ for automated modeling of a continuous system

Subroutine B1END is the last of a set of subroutines used for modeling the connection of a set of s plane transfer functions for automated transfer function analysis of a continuous system. It checks to make sure that all the C_i blocks have been properly specified. The connection of the C_i blocks specified by the calls to subroutine B1CEQ is printed out.

FORTRAN CALL CALL B1END( )

PRECMP DIRECTIVE *B1END

produces the following FORTRAN statement

```
CALL B1END( )
```

COMMENTS If a C_i block has not been properly specified the program will abort.

PURPOSE: Compute SISO frequency response between blocks $\mathbf{C}_i$ and $\mathbf{C}_j$ of a continuous system modeled as a connection of s plane transfer functions

A SISO frequency response of an s plane transfer function is computed for a continuous system modeled as a connection[1] of s plane transfer functions. The continuous connection block $\mathbf{C}_i$ which the input is connected to is specified by parameter UCIN. The magnitude of the input is specified by the parameter UMAGN. The continuous connection block $\mathbf{C}_j$ which defines the output is specified by parameter YCOUT. Parameter PRNMTRX is used to specify the type of output for the state space matrices computed by this command. These parameters are in common block DBASE described in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
UCIN	0	$\mathbf{C}_i$ block number where input u is connected to
UMAGN	1.	Magnitude of the input u
YCOUT	0	$\mathbf{C}_i$ block number defining the output
PRNMTRX	0	.EQ.0 No printout of state space matrices .EQ.1 Printout of computed state space matrices .EQ.2 Above plus intermediate matrices

This command differs from command SFREQ which computes the frequency response of an s plane transfer function $\mathbf{SPTF}_i$ represented in rational function. Command B1FREQ computes the frequency response of a transfer function directly from a state space representation of a system modeled by a connection of $\mathbf{SPTF}_i$ transfer functions. It uses Laub's method which does not involve the computation of the poles and zeros of the transfer function.

Command B1FREQ is an alternate method for computing the frequency response of a system modeled as a connection of $\mathbf{SPTF}_i$ transfer functions. The combination of commands B1TF and SFREQ will yield the poles and zeros as well as the frequency response of a transfer function. As discussed in Section 7.2, computation of the poles and zeros is a difficult numerical problem to solve in a reliable and accurate manner. If the accuracy of the poles or zeros from a B1TF command are in question, command B1FREQ can be used to check the frequency response of a transfer function computed by the combination of commands B1TF and SFREQ.

Two modes are available for selecting the frequency points used for evaluating the frequency response of the transfer function. The automatic mode, selected when FAUTO is nonzero, allows the user to specify up to 20 preselected frequencies with the array OMEGA. In this mode the program will use all these preselected points plus its own dynamically generated points to yield a smooth plot. The number of preselected points is determined by NOMEGA. The beginning and last frequency points for computing the response are determined by OMEGA(1) and OMEGA(NOMEGA), respectively.

[1] With calls to subroutines B1INIT, B1CEQ, and B1END.

In the nonautomatic frequency mode (FAUTO=0) the user can define up to five sets of frequencies to be used in computing the response. Each of these sets is specified by a three element array of the form FREQk(i), i=1,3. If FREQk(1)=a, FREQk(2)=b and FREQk(3)=c, the k-th set of frequencies specified is:

a, a+c, a+2c, ... a+jc, b

where j is the largest integer such that (a+jc) is less than b. Each successive FREQk array must define an increasing set of frequencies such that the first value of the segment is always larger than the last value of the preceding segment. When FREQk(3) is not larger than FREQk(1), as in the case with the preset values for $k = 2, 5$, those segments will not be used.

With either the automatic or the nonautomatic frequency mode, the program will automatically check for the gain and phase crossover. When found, the program will iterate until the exact cross over frequency is found. The limit on the number of plot points computed is 1500.

The units used for representing transfer functions are in rad/sec. For frequency response calculations, the frequencies used for evaluating the response can either be in rad/sec or Hz. A nonzero value for the parameter RAD will select the units rad/sec, while a zero value will select Hz.

Bode, Nichols, and Nyquist plots are selected by the values of the flags FBODE, FNICO, FNYQS, respectively. See the table in the Reference for command SFREQ for the definition and default values of parameters used by command B1FREQ.

FORTRAN CALL CALL B1FREQ( )

Example 1: Assume that a system has been modeled as a connection of **SPTF**$_i$ transfer functions with calls to subroutines B1INIT, B1CEQ, and B1END. Compute the frequency response of the transfer function from block $\mathbf{C}_{13}$ to block $\mathbf{C}_2$ using the auto frequency mode for frequencies between 1. and 100. and at 10. Select printer and hardcopy Bode plots with auto scaling. The FORTRAN code can be written as:

```
UCIN=13
YCOUT=2
UMAGN=1.
FAUTO=1
NOMEGA=3
OMEGA(1)=1.
OMEGA(2)=10.
OMEGA(3)=100.
FBODE=1
GRAFP=1
HRDCPY=1
CALL B1FREQ( )
```

PRECMP DIRECTIVE *B1FREQ

produces the following FORTRAN statement

```
CALL B1FREQ( )
```

Example 2: Using Example 1 from above, the following code fragment

```
      UCIN=13
      YCOUT=2
      UMAGN=1.
      FAUTO=1
      NOMEGA=3
*OMEGA 1 10 100
      FBODE=1
      GRAFP=1
      HRDCPY=1
*B1FREQ
```

produces the following FORTRAN statements

```
      UCIN=13
      YCOUT=2
      UMAGN=1.
      FAUTO=1
      NOMEGA=3
      OMEGA(1)=10.
      OMEGA(2)=10.
      OMEGA(3)=100.
      FBODE=1
      GRAFP=1
      HRDCPY=1
      CALL B1FREQ( )
```

METHOD Section 7.2 describes how a state space representation is computed for a system modeled by a connection of **SPTF**$_i$ transfer functions. See the Reference for command SFREQ on how the frequency points are automatically selected if the parameter FAUTO is nonzero.

If a frequency point being used to evaluate the response is equal to a pole on the $j\omega$ axis, the warning message

(S*I - A) IS SINGULAR TO WORKING PRECISION AT OMEGA = ..., THIS POINT SKIPPED

will be printed out and the program allowed to continue.

PURPOSE: Initialization routine for automated modeling of a continuous system

Subroutine B1INIT is the first of a set of subroutines used for modeling the connection of s plane transfer function blocks for automated transfer function analysis of a continuous system. This initialization is used primarily to let the program know that a set of calls to subroutine B1CEQ will follow which will specify the connection of each $\mathbf{C}_i$ block.

FORTRAN CALL `CALL B1INIT( tlabel, oldnew, ncblk )`

where, tlabel = label for block 1 connection (.LE.60 characters)
oldnew = 'OLD' for old data, (no initialization)
= 'NEW' for new data, (initializes all connections)
ncblk = number of s plane connection blocks (.LE.MDIMYC)

Example 1: A continuous system with 7 s plane transfer function blocks are to be modeled for automated transfer function analysis. The model is to be labeled as 'PROJECT XYZ'. The FORTRAN code for initializing the connection of this model can be written as:

```
 CALL B1INIT('PROJECT XYZ','NEW',7)
```

PRECMP DIRECTIVE `*B1INIT tlabel oldnew ncblk`

produces the following FORTRAN statements

```
 CALL B1INIT(
+tlabel
+,oldnew,ncblk)
```

Example 2: Using Example 1 from above, the following code fragment

```
*B1INIT 'PROJECT XYZ' 'NEW' 7
```

produces the following FORTRAN statement

```
 CALL B1INIT('PROJECT XYZ'
+,'NEW',7)
```

COMMENTS See Chapter 7.

PURPOSE: Load a file of block diagram connection data previously saved with the command B1SAVE

FORTRAN CALL CALL B1LOAD(file_name)

where, file_name = local file name (.LE.7 characters)
(the local file must be attached before execution of a batch job)

Example 1: Load transfer function connection data for modeling a continuous system from local file B1DATA. The required FORTRAN code can be written as:

```
CALL B1LOAD('B1DATA')
```

PRECMP DIRECTIVE *B1LOAD file_name

produces the following FORTRAN statement

```
CALL B1LOAD(file_name)
```

Example 2: `*B1LOAD 'B1DATA'`

produces the following FORTRAN statement

```
CALL B1LOAD('B1DATA')
```

COMMENTS See the Reference for command B1SAVE for description of the file.

PURPOSE: Automated root loci of a continuous system modeled as a connection of s plane transfer functions

Root locus is automatically computed for a continuous system modeled as a connection[1] of s plane transfer functions. This is a more general root loci command than SLOCI which computes the root locus by varying the loop gain of a $\mathbf{SPTF}_i$ transfer function. With the B1LOCI command, the root locus is computed by varying the gain of any $\mathbf{C}_i$ block. The gains to be used are multipliers of the transfer function associated with the $\mathbf{C}_i$ block selected. Thus, the gains should begin at a value less than 1.0 and end with a value greater than 1.0 if the loci is to bracket the nominal operating gain of the system. The method of specifying the gains to be used in computing the root loci is identical to that used by command SLOCI. The printout of the state space matrices computed by this command is specified by the parameter PRNMTRX in common block DBASE described by the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
PRNMTRX	0	.EQ.0 No printout of state space matrices .EQ.1 Printout of computed state space matrices .EQ.2 Above plus intermediate matrices

Up to 25 preselected gains for evaluating the root loci can be specified with array KGAIN. The number of preselected gains is determined by NLOCI. The beginning and last gains are determined by KGAIN(1) and KGAIN(NLOCI), respectively. Additional gains can be automatically computed by the program to fill in values between the preselected gains. Between KGAIN(i) and KGAIN(i+1), additional gains will be selected by either of the following two methods if they are between KGAIN(i) and KGAIN(i+1).

If KFLG.EQ.0, gain = KGAIN(i) + KDELT*j, j=1,2, ...

If KFLG.NE.0, gain = KGAIN(i)*KDELT**j, j=1,2, ...

For example, if KGAIN(2)=10, KGAIN(3)=100, KFLG=1, and KDELT=2, the following gains will be used to compute the root loci:

. . . 10., 20., 40., 80., 100., . . .

The total number of gains used is limited by the value of the parameter ITLOC which is preset to 50. Parameters used by B1LOCI are given in the following table.

[1] Specified with calls to subroutines B1INIT, B1CEQ, and B1END.

COMMON/DBASE/ parameters used		
parameter	preset	description
NLOCI	2	Number of root locus gains entered in array KGAIN (max=25)
KGAIN	 .5 2.	Array of root locus gains KGAIN(1) = first user-specified root locus gain KGAIN(2) = second user-specified root locus gain KGAIN(NLOCI) = last user-specified root locus gain. (Gains computed and used only if they are between KGAIN(1) and KGAIN(NLOCI))
KFLG	1	.EQ.0 to increment gain by multiplying by KDELT .NE.0 to increment gain by adding by KDELT
KDELT	1.E4	Value for changing gains (preset to large value so that no additional gains are computed).
ITLOC	50	Max. number of different gains computed
GRAFP	1	.NE.0 for low resolution printer plots
HRDCPY	0	.NE.0 for high resolution electrostatic plots
RLXMIN	0.	Min. x axis for plotting
RLXMAX	0.	Max. x axis for plotting (Auto. scaling of x axis if RLXMIN=RLXMAX)
RLYMIN	0.	Min. y axis for plotting
RLYMAX	0.	Max. y axis for plotting (Auto. scaling of y axis if RLYMIN=RLYMAX)
RLFLG1	1	Flag for numbering root locus points on hardcopy plots .EQ.1 for numbering; .EQ.-1 for no numbering

FORTRAN CALL CALL B1LOCI(indx)

where, indx = block identifier of $\mathbf{C}_{\text{indx}}$ block where gain is to be varied (.LE. number of continuous connection blocks specified in call to B1INIT)

Example 1: Compute the the root locus of continuous system modeled as a connection of transfer function blocks by varying the gain of block $\mathbf{C}_4$ from 2 to 20 by doubling the first gain until the last gain is reached. Select printer and hardcopy plots with auto scaling. The FORTRAN code can be written as:

```
NLOCI=2
KGAIN(1)=2
KGAIN(2)=20
KFLG=0
KDELT=2.
GRAFP=1
HRDCPY=1
CALL B1LOCI(4)
```

PRECMP DIRECTIVE *B1LOCI indx

produces the following FORTRAN statement

```
CALL B1LOCI(indx)
```

Example 2:

```
*B1LOCI 4
```

produces the following FORTRAN statement

```
CALL B1TF(4)
```

METHOD See Chapter 7.

COMMENTS The connection of the transfer function blocks must represent the closed loop configuration of the system.

PURPOSE: Save the transfer function connection data for modeling a continuous systems to a local file

This subroutine will save the transfer function connection data[1] for modeling a continuous system. The data will be saved to a local file which the user can reuse in the same job by reloading it with the B1LOAD command. If the file is to be used in a subsequent batch job or an interactive session, it must be SAVEd (CRAY) or CATALOGed (CDC) upon exit from LCAP2.

FORTRAN CALL CALL B1SAVE(file_name)

where, file_name = local file name (.LE.7 characters)

Example 1: Store the transfer function connection data for modeling a continuous system in a local file named B1DATA. The FORTRAN code can be written as:

```
CALL B1SAVE('B1DATA')
```

PRECMP DIRECTIVE *B1SAVE file_name

produces the following FORTRAN statement

```
CALL B1SAVE(file_name)
```

Example 2:

```
*B1SAVE 'B1DATA'
```

produces the following FORTRAN statement

```
CALL B1SAVE('B1DATA')
```

METHOD The transfer function connection data of a continuous system is converted to an ASCII file consisting of a set of PRECMP compatible directives. This allows the file to be easily read and modified, if necessary, by an editor. In addition to being read by command B1LOAD, this file can also be read by the PRECMP precompiler. This feature can be utilized in conjunction with interactive LCAP2 to simplify the effort required to model the connection of transfer functions for a batch job.

Instead of using the subroutines B1INIT, B1CEQ, and B1END to specify the transfer function connection of a continuous system in a batch job, the user can use command B1CONN in interactive LCAP2 instead. This command is much easier to use since it is prompt driven and includes logic to test for reasonable inputs. After verifying that the model is correctly specified, the command B1SAVE is used to save the connection data. This file will then contain the necessary PRECMP

[1] Specified by calls to subroutines B1INIT, B1CEQ, and B1END.

directives which can be used as code for a batch job. Is should be pointed out that the PRECMP directives in the file produced by the B1SAVE command contains only numeric values whereas the general PRECMP directives allow FORTRAN expressions. This procedure of using interactive LCAP2 to develop the PRECMP code for modeling the connection of transfer functions can save the user significant setup time. Once this code is incorporated into a batch job, the PRECMP code can be changed to include FORTRAN variables or expressions to make the program more flexible.

PURPOSE: Compute SISO transfer function between blocks $\mathbf{C}_i$ and $\mathbf{C}_j$ of a continuous system modeled as a connection of s plane transfer functions
SPTF$_i$ = SISO transfer function evaluated

A SISO transfer function is automatically evaluated for a continuous system modeled[1] as a connection of s plane transfer functions. The continuous connection block $\mathbf{C}_i$ which the input is connected to is specified by parameter UCIN. The magnitude of the input is specified by the parameter UMAGN. The continuous connection block $\mathbf{C}_j$ which defines the output is specified by the parameter YCOUT. Parameter PRNMTRX is used to specify the type of output for the state space matrices computed by this command. These parameters are in the common block DBASE described in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
UCIN	0	$\mathbf{C}_i$ block number where input u is connected to
UMAGN	1.	Magnitude of the input u
YCOUT	0	$\mathbf{C}_i$ block number defining the output
PRNMTRX	0	.EQ.0 No printout of state space matrices .EQ.1 Printout of computed state space matrices .EQ.2 Above plus intermediate matrices

FORTRAN CALL CALL B1TF(i)

where, i = index for storing evaluated transfer function **SPTF**$_i$

Example 1: Compute the transfer function from $\mathbf{C}_3$ to $\mathbf{C}_6$ for a unit input. Store the resulting transfer function in **SPTF**$_2$. The FORTRAN code can be written as:

```
UCIN=3
YCOUT=6
UMAGN=1.
CALL B1TF(2)
```

PRECMP DIRECTIVE *B1TF i

produces the following FORTRAN statement

```
CALL B1TF(i)
```

[1] Specified by calls to subroutines B1INT, B1CEQ, and B1END.

Example 2:

```
*B1TF 2
```

produces the following FORTRAN statement

```
 CALL B1TF(2)
```

METHOD State space method is used to model the system. The QR and QZ methods are used to find, respectively, the poles and zeros of the transfer function. See Chapter 7 and Section C.2.

PURPOSE: Compute SISO time response between blocks C_i and C_j of a continuous system modeled as a connection of s plane transfer functions and sample-hold blocks

An SISO time response is automatically evaluated for a continuous system modeled as a connection[1] of s plane transfer functions. The continuous connection block C_i which the input is connected to is specified by parameter UCIN. The type of input and its magnitude is specified, respectively, by parameters TTYPE and UMAGN. The continuous connection block C_j which defines the output is specified by parameter YCOUT. The response will be evaluated every TDELT seconds from t=0 to t=TEND. Parameter PRNMTRX is used to specify the type of output for the state space matrices computed by this command. Parameters used by B1TIME are in common block DBASE described in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
UCIN	0	C_i block number where input u is connected to
TTYPE	1	.EQ.0 for impulse response; .EQ.1 for step response
UMAGN	1.	Magnitude of input
YCOUT	0	C_i block number defining the output
TDELT	1.	Delta time used for evaluating continuous time response
TEND	1.	End time for evaluating time response
PRNMTRX	0	.EQ.0 No printout of state space matrices .EQ.1 Printout of computed state space matrices .EQ.2 More printout of computed state space matrices
GRAFP	1	.NE.0 for low resolution printer plots
HRDCPY	0	.NE.0 for high resolution electrostatic plots
TXMIN	0.	Minimum x axis for time plot
TXMAX	0.	Maximum x axis for time plot (Auto scaling of x axis if TXMIN=TXMAX)
TYMIN	0.	Minimum y axis for time plot
TYMAX	0.	Maximum y axis for time plot (Auto scaling of y axis if TYMIN=TYMAX)
CONTP	0	=0 Single curve plot =1 First curve of a plot =2 Continuation of a plot =3 Final curve of a plot

FORTRAN CALL CALL B1TIME( )

[1] Specified by calls to subroutines B1INIT, B1CEQ, and B1END.

Example 1: Assume that a system has been modeled as a connection of **SPTF**$_i$ transfer functions with calls to subroutines B1INIT, B1CEQ, and B1END. Compute the step response from block $\mathbf{C}_3$ to block $\mathbf{C}_7$ from t=0 to t=5 seconds in increments of .2 seconds. Select only a high resolution plot.

The FORTRAN code can be written as:

```
UCIN=3
TTYPE=1
UMAGN=1.
YDOUT=7
TDELT=.2
TEND=5.
GRAFP=0
HRDCPY=1
CALL B1TIME( )
```

PRECMP DIRECTIVE *B1TIME

produces the following FORTRAN statements

```
CALL B1TIME( )
```

METHOD State space method, see Section 7.2.1. Unlike command STIME, this command does not have any restrictions on multiple poles.

COMMENTS Capability exists for creating multiple plots on the high resolution electrostatic plotter by setting the appropriate value for the parameter CONTP. See Section E.3 for the details.

PURPOSE: Equation definition for a $\mathbf{C}_i$ continuous block used for automated modeling of a continuous-discrete multirate system by connection of transfer function blocks

Subroutine B2CEQ is part of a set of subroutines used for defining the connection of s plane transfer functions for automated transfer function analysis of a continuous-discrete multirate system. (See B2DEQ and B2SEQ for defining the discrete and sample-hold blocks). This subroutine is to be called for each $\mathbf{C}_i$ block defined. Before calling B2CEQ, B2DEQ, and B2SEQ, the initialization routine B2INIT must first be called. After the last $\mathbf{C}_i$, $\mathbf{D}_i$, and $\mathbf{S}_i$ blocks are defined by a calls to subroutines B2CEQ, B2DEQ, and B2SEQ, respectively, subroutine B2END must be called to complete the connection procedure.

FORTRAN CALL

```
CALL B2CEQ( label, indx, isptf, delay, iyin, nycin, nxsin, iycin, ixsin )
```

where, label = label for block $\mathbf{C}_i$ (.LE.60 characters)
indx = block identifier for block $\mathbf{C}_i$ (.LE.NDIMYC)
isptf = **SPTF** identifier for block $\mathbf{C}_i$
delay = delay of block $\mathbf{C}_i$, not implemented yet, but must enter a dummy value
iyin = for future use. Enter a dummy value.
nycin = number of continuous blocks connected as inputs to block $\mathbf{C}_i$
nxsin = number of sample-hold blocks connected as inputs to block $\mathbf{C}_i$
iycin = array containing the identifiers of the continuous blocks connected to $\mathbf{C}_i$. (negative value for sign change)
ixsin = array containing the identifiers of the sample-hold blocks connected to $\mathbf{C}_i$. (negative value for sign change)

The dummy arrays iycin and ixsin can be any integer array dimensioned at least nycin and nxsin, respectively, which the user can declare. For convenience, though, arrays IYCIN and IXSIN in common block IYCDS can be used instead. These arrays are described in the following table.

COMMON/IYCDS/ parameters used		
parameter	preset	description
IYCIN	0	Array containing the identifiers of the continuous blocks connected to $\mathbf{C}_i$ IYCIN(1) = id number of 1st continuous block connected to $\mathbf{C}_i$ IYCIN(2) = id number of 2nd continuous block connected to $\mathbf{C}_i$ ⋮ IYCIN(nycin) = id number of nycin-th continuous block connected to $\mathbf{C}_i$
IXSIN	0	Array containing the identifiers of the sample-hold blocks connected to $\mathbf{C}_i$ IXSIN(1) = id number of 1st sample-hold block connected to $\mathbf{C}_i$ IXSIN(2) = id number of 2nd sample-hold block connected to $\mathbf{C}_i$ ⋮ IXSIN(nxsin) = id number of nxsin-th sample-hold block connected to $\mathbf{C}_i$

Example 1: Block $\mathbf{C}_7$ is to be represented by s plane transfer $\mathbf{SPTF}_4$. The are three continuous blocks, $\mathbf{C}_1$, $\mathbf{C}_5$, and $\mathbf{C}_6$ and two sample-hold blocks, $\mathbf{S}_2$ and $\mathbf{S}_4$ connected to the input of block $\mathbf{C}_3$. Block $\mathbf{C}_3$ is to be labeled as 'FILTER XX'. The FORTRAN code to define block $\mathbf{C}_3$ can be written as:

```
 INDX=7
 ISPTF=4
 DELAY=0
 NYCIN=3
 IYCIN(1)=1
 IYCIN(2)=5
 IYCIN(3)=6
 NXSIN=2
 IXSIN(1)=2
 IXSIN(2)=4
 CALL B2CEQ('FILTER XX',INDX,ISPTF,DELAY,0,NYCIN,NXSIN,IYCIN
+,IXSIN)
```

PRECMP DIRECTIVE *B2CEQ label indx isptf delay iyin nycin nxsin

produces the following FORTRAN statements

```
 CALL B2CEQ  (
+label
+,indx,isptf,delay,iyin,nycin,nxsin,IYCIN,IXSIN)
```

Note that IYCIN and IXSIN are not arguments in the B2CEQ directive.

Example 2: Using Example 1 from above, the following code fragment

```
*IYCIN 1 5 6
*IXSIN 2 4
*B2CEQ 'FILTER XX' 7 4 0. 0 3 2
```

produces the following FORTRAN statements

```
 IYCIN(1)=1
 IYCIN(2)=5
 IYCIN(3)=6
 IXSIN(1)=2
 IXSIN(2)=4
 CALL B2CEQ(
+'FILTER XX'
+,7,4,0.,0,3,2,IYCIN,IXSIN)
```

METHOD The Kalman-Bertram state space method is used. See Chapter 7.

COMMENTS See Example 18 in Chapter 9.

PURPOSE: Equation definition for a $\mathbf{D}_i$ discrete block used for automated modeling of a continuous-discrete multirate system by connection of transfer function blocks

Subroutine B2DEQ is part of a set of subroutines used for defining the connection of z plane transfer functions for automated transfer function analysis of a continuous-discrete multirate system. (See B2CEQ and B2SEQ for defining the continuous and sample-hold blocks). This subroutine is to be called for each $\mathbf{D}_i$ block defined. Before calling B2CEQ, B2DEQ, and B2SEQ, the initialization routine B2INIT must first be called. After the last $\mathbf{C}_i$, $\mathbf{D}_i$, and $\mathbf{S}_i$ blocks are defined by a calls to subroutines B2CEQ, B2DEQ, and B2SEQ, respectively, subroutine B2END must be called to complete the connection procedure.

FORTRAN CALL

```
 CALL B2DEQ( label, indx, izptf, dsampt, delay, iyin, nycin, nydin
+,nxsin, iycin, iydin, ixsin )
```

where, label = label for block $\mathbf{D}_i$ (.LE.60 characters)
indx = block identifier for block $\mathbf{D}_i$ (.LE.NDIMYD)
izptf = **ZPTF** identifier for block $\mathbf{D}_i$
dsampt = sampling period of block $\mathbf{D}_i$
delay = delay of block $\mathbf{D}_i$, not implemented yet, but must enter a dummy value
iyin = for future use. Enter a dummy value.
nycin = number of continuous blocks connected as inputs to block $\mathbf{D}_i$
nydin = number of discrete blocks connected as inputs to block $\mathbf{D}_i$
nxsin = number of sample-hold blocks connected as inputs to block $\mathbf{D}_i$
iycin = array containing the identifiers of the continuous blocks connected to $\mathbf{D}_i$. (negative value for sign change)
iydin = array containing the identifiers of the discrete blocks connected to $\mathbf{D}_i$. (negative value for sign change)
ixsin = array containing the identifiers of the sample-hold blocks connected to $\mathbf{D}_i$. (negative value for sign change)

The dummy arrays iycin, iydin, and ixsin can be any integer array dimensioned at least nycin, nydin, and nxsin, respectively, which the user can declare. For convenience, though, arrays IYCIN, IYDIN, and IXSIN in common block IYCDS can be used instead. These arrays are described in the following table.

COMMON/IYCDS/ parameters used		
parameter	preset	description
IYCIN	0	Array containing the identifiers of the continuous blocks connected to $\mathbf{D}_i$ IYCIN(1) = id number of 1st continuous block connected to $\mathbf{D}_i$ IYCIN(2) = id number of 2nd continuous block connected to $\mathbf{D}_i$ ⋮ IYCIN(nycin) = id number of nycin-th continuous block connected to $\mathbf{D}_i$
IYDIN	0	Array containing the identifiers of the discrete blocks connected to $\mathbf{D}_i$ IYDIN(1) = id number of 1st discrete block connected to $\mathbf{D}_i$ IYDIN(2) = id number of 2nd discrete block connected to $\mathbf{D}_i$ ⋮ IYDIN(nydin) = id number of nydin-th discrete block connected to $\mathbf{D}_i$
IXSIN	0	Array containing the identifiers of the sample-hold blocks connected to $\mathbf{D}_i$ IXSIN(1) = id number of 1st sample-hold block connected to $\mathbf{D}_i$ IXSIN(2) = id number of 2nd sample-hold block connected to $\mathbf{D}_i$ ⋮ IXSIN(nxsin) = id number of nxsin-th sample-hold block connected to $\mathbf{D}_i$

Example 1: Block $\mathbf{D}_8$ is to be represented by z plane transfer $\mathbf{ZPTF}_{10}$ and its sampling period is .5 seconds. There is one continuous blocks $\mathbf{C}_4$, two discrete blocks, $\mathbf{D}_3$, and $\mathbf{D}_5$, and one sample-hold block $\mathbf{S}_2$ connected to the input of block $\mathbf{D}_8$. Block $\mathbf{C}_3$ is to be labeled as 'DIGITAL FILTER A, VERSION 3'. The FORTRAN code to define block $\mathbf{D}_8$ can be written as:

```
INDX=8
IZPTF=10
DSAMPT=.5
DELAY=0
NYCIN=1
IYCIN(1)=4
NYDIN=2
IYDIN(1)=3
IYDIN(2)=5
NXSIN=1
IXSIN(1)=2
CALL B2DEQ('DIGITAL FILTER A, VERSION 3',INDX,IZPTF,DSAMPT,DELAY
+,0,NYCIN,NYDIN,NXSIN,IYCIN,IYDIN,IXSIN)
```

PRECMP DIRECTIVE *B2DEQ label indx izptf dsampt delay iyin nycin nydin nxsin

produces the following FORTRAN statements

```
 CALL B2CEQ  (
+label
+,indx,izptf,dsampt,delay,iyin,nycin,nydin,nxsin,IYCIN
+,IYDIN,IXSIN)
```

Note that IYCIN, IYDIN, and IXSIN are not arguments in the *B2DEQ directive.

Example 2: Using Example 1 from above, the following code fragment

```
*IYCIN 4
*IYDIN 3 5
*IXSIN 2
      DSAMPT=.5
*B2DEQ 'DIGITAL FILTER A, VERSION 3' ..  note use of continuation
 8 10 DSAMPT 0. 0 1 2 1 ! character above and comment character !
```

produces the following FORTRAN statements

```
 IYCIN(1)=4
 IYDIN(1)=3
 IYDIN(2)=5
 IXSIN(1)=2
 CALL B2DEQ(
+'DIGITAL FILTER A, VERSION 3'
+,8,10,DSAMPT,0.,0,1,2,1,IYCIN,IYDIN,IXSIN)
```

METHOD The Kalman-Bertram state space method is used. See Chapter 7 and Section C.2.

COMMENTS Part of the automated procedure for implementing the Kalman-Bertram method of analysis is computing the LCM sampling period of the system. In specifying the sampling periods of each z transfer function or sample-hold block, be sure to use enough significant digits. Failure to do so can result in an incorrect LCM calculation. For example, if the sampling rates are 1, 1/3, and 2/3, enter the sampling rates as 1., .3333333, and .6666666. If fewer digits were used instead, i.e., .3333 instead of .3333333, the LCM will be incorrectly computed.

See Example 18 in Chapter 9.

PURPOSE: Compute eigenvalues of a continuous-discrete multirate system modeled as a connection of s and z plane transfer functions and sample-hold blocks
$\mathbf{POLY}_i$ = eigenvalues of system matrix, equivalent to det [zI - A]

The eigenvalues of a continuous-discrete system modeled as a connection[1] of s and z plane transfer functions and sample-hold blocks are computed. The z plane eigenvalues computed are at the LCM sampling period.[2] The printout of the state space matrices computed by this command is specified by the parameter PRNMTRX in common block DBASE described by the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
PRNMTRX	0	.EQ.0 No printout of state space matrices .EQ.1 Printout of computed state space matrices .EQ.2 Above plus intermediate matrices

FORTRAN CALL CALL B2EIG(i)

where, i = index for storing polynomial $\mathbf{POLY}_i$

Example 1: Compute the eigenvalues of a continuous-discrete system modeled as a connection of s and z plane transfer functions and sample-hold blocks. Store the results into $\mathbf{POLY}_2$. The FORTRAN code can be written as:

```
CALL B2EIG(2)
```

PRECMP DIRECTIVE *B2EIG i

produces the following FORTRAN statement

```
CALL B2EIG(i)
```

Example 2:

```
*B2EIG 2
```

produces the following FORTRAN statement

```
CALL B2EIG(2)
```

METHOD The Kalman-Bertram state space method is used. See Chapter 7.

[1] Specified by calls to subroutines B2INIT, B2CEQ, B2DEQ, B2SEQ, and B2END.

[2] Least-common-multiple of all the sampling periods of the $\mathbf{D}_i$ and $\mathbf{S}_i$ connection blocks.

PURPOSE: Complete the transfer function connection procedure initiated by calls to subroutines B2INIT, B2CEQ, B2DEQ, and B2SEQ for automated modeling of a continuous-discrete multirate system

Subroutine B2END is the last of a set of subroutines used for modeling the connection of a set of (1) s plane transfer functions, (2) z plane transfer functions, and (3) sample-hold blocks for automated transfer function analysis of a continuous-discrete multirate system. It checks to make sure that all the $\mathbf{C}_i$, $\mathbf{D}_i$, and $\mathbf{S}_i$ blocks have been properly specified. The connection of the $\mathbf{C}_i$, $\mathbf{D}_i$, and $\mathbf{S}_i$ blocks specified by the calls to subroutines B2CEQ, B2DEQ, and B2SEQ, respectively, is printed out.

FORTRAN CALL CALL B2END( )

PRECMP DIRECTIVE *B2END

produces the following FORTRAN statement

```
CALL B2END( )
```

COMMENTS If a $\mathbf{C}_i$, $\mathbf{D}_i$, or $\mathbf{S}_i$ block has not been properly specified the program will abort.

PURPOSE: Compute SISO frequency response between blocks $\mathbf{D}_i$ and $\mathbf{D}_j$ of a continuous-discrete multirate system modeled as a connection of s and z plane transfer functions and sample-hold blocks

A SISO frequency response of a z plane transfer function is computed for a continuous-discrete multirate system modeled as a connection[1] of s and z plane transfer functions and sample-hold blocks. The discrete connection block $\mathbf{D}_i$ which the input is connected to is specified by parameter UDIN. The magnitude of the input is specified by the parameter UMAGN. The discrete connection block $\mathbf{D}_j$ which defines the the output is specified by the parameter YDOUT. The sampling period of this transfer function will be the computed LCM sampling period.[2] Parameter PRNMTRX is used to to specify the type of output for the state space matrices computed by this command. These parameters are in common block DBASE described in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
UDIN	0	$\mathbf{D}_i$ block number where input u is connected to
UMAGN	1.	Magnitude of the input u
YDOUT	0	$\mathbf{D}_i$ block number defining the output
PRNMTRX	0	.EQ.0 No printout of state space matrices .EQ.1 Printout of computed state space matrices .EQ.2 Above plus intermediate matrices

This command differs from command ZFREQ which computes the frequency response of a z plane transfer function $\mathbf{ZPTF}_i$ represented in rational function. Command B2FREQ computes the frequency response of a transfer function directly from a state space representation of a system modeled by a connection of $\mathbf{SPTF}_i$ and $\mathbf{ZPTF}_i$ transfer functions. It uses Laub's method which does not involve the computation of the poles and zeros of the transfer function.

Command B2FREQ is an alternate method for computing the frequency response of a system modeled as a connection of $\mathbf{SPTF}_i$ and $\mathbf{ZPTF}_i$ transfer functions. The combination of commands B2TF and ZFREQ will yield the poles and zeros as well as the frequency response of a transfer function. As discussed in Section 7.2, computation of the poles and zeros is a difficult numerical problem to solve in a reliable and accurate manner. If the accuracy of the poles or zeros from a B2TF command are in question, command B2FREQ can be used to check the frequency response of a transfer function computed by the combination of commands B2TF and ZFREQ.

Two modes are available for selecting the frequency points used for evaluating the frequency response of the transfer function. The automatic mode, selected when FAUTO is nonzero, allows the user to specify up to 20 preselected frequencies with the array OMEGA. In this mode the program will use all these preselected points plus its own dynamically generated points to yield a smooth

[1] With calls to subroutines B2INIT, B2CEQ, B2DEQ, B2SEQ, and B2END

[2] Lowest-common-multiple of the sampling periods of the $\mathbf{D}_i$ and $\mathbf{S}_i$ blocks (which is computed by the program from data entered in the calls to B2DEQ and B2SEQ).

plot. The number of preselected points is determined by NOMEGA. The beginning and last frequency points for computing the response is determined by OMEGA(1) and OMEGA(NOMEGA), respectively.

In the nonautomatic frequency mode (FAUTO=0) the user can define up to five sets of frequencies to be used in computing the response. Each of these sets is specified by a three element array of the form FREQk(i), i=1,3. If FREQk(1)=a, FREQk(2)=b and FREQk(3)=c, the k-th set of frequencies specified is:

$$a, a+c, a+2c, \ldots\ a+jc, b$$

where j is the largest integer such that (a+jc) is less than b. Each successive FREQk array must define an increasing set of frequencies such that the first value of the segment is always larger than the last value of the preceding segment. When FREQk(3) is not larger than FREQk(1), as in the case with the preset values for $k = 2, 5$, those segments will not be used.

With either the automatic or the nonautomatic frequency mode, the program will automatically check for the gain and phase crossover. When found, the program will iterate until the exact crossover frequency is found. The limit on the number of plot points computed is 1500.

The units used for representing transfer functions are in rad/sec. For frequency response calculations, the frequencies used for evaluating the response can either be in rad/sec or Hz. A nonzero value for the parameter RAD will select the units rad/sec, while a zero value will select Hz.

Bode, Nichols, and Nyquist plots are selected by the values of the flags FBODE, FNICO, FNYQS, respectively. See the table in the Reference for command ZFREQ for the definition and default values of parameters used by command B2FREQ.

FORTRAN CALL CALL B2FREQ( )

Example 1: Assume that a system has been modeled as a connection of **SPTF**$_i$ and **ZPTF**$_i$ transfer functions with calls to subroutines B2INIT, B2CEQ, B2DEQ, B2SEQ and B2END. Compute the frequency response of the transfer function from block **D**$_3$ to block **D**$_7$ using the auto frequency mode for frequencies between 1. and 100. and at 10. Select printer and hardcopy Bode plots with auto scaling. The FORTRAN code can be written as:

```
UDIN=3
YDOUT=7
UMAGN=1.
FAUTO=1
NOMEGA=3
OMEGA(1)=1.
OMEGA(2)=10.
OMEGA(3)=100.
FBODE=1
GRAFP=1
HRDCPY=1
CALL B2FREQ( )
```

PRECMP DIRECTIVE $\boxed{\text{*B2FREQ}}$

produces the following FORTRAN statement

```
CALL B2FREQ( )
```

Example 2: Using Example 1 from above, the following code fragment

```
      UDIN=3
      YDOUT=7
      UMAGN=1.
      FAUTO=1
      NOMEGA=3
*OMEGA 1 10 100
      FBODE=1
      GRAFP=1
      HRDCPY=1
*B2FREQ
```

produces the same FORTRAN code as in Example 1.

METHOD Section 7.2 describes how a state space representation is computed for a system modeled by a connection of **SPTF**$_i$ and **ZPTF**$_i$ transfer functions. See the Reference for command ZFREQ on how the frequency points are automatically selected if the parameter FAUTO is nonzero.

If a frequency point being used to evaluate the response is equal to a pole on the unit circle, the warning message

(z*I - A) IS SINGULAR TO WORKING PRECISION AT OMEGA = ..., THIS POINT SKIPPED

will be printed out and the program allowed to continue.

COMMENTS The sampling periods for both the discrete connection blocks used to define the input and output of the transfer function must be equal to the computed LCM sampling period. (Note: The parameter SAMPT is not by this command.)

PURPOSE: Initialization routine for automated modeling of a continuous-discrete multirate system

Subroutine B2INIT is the first of a set of subroutines used for modeling the connection of s plane transfer functions, z plane transfer functions, and sample-hold blocks for automated transfer function analysis of a continuous-discrete multirate system. This initialization is used primarily to let the program know that a set of calls to subroutines B2CEQ, B2DEQ, and B2SEQ will follow which will specify the connection of each $\mathbf{C}_i$, $\mathbf{D}_i$, and $\mathbf{S}_i$ block.

FORTRAN CALL CALL B2INIT(tlabel, oldnew, ncblk, ndblk, nshblk)

where, tlabel = label for block 2 connection (.LE.60 characters)
oldnew = 'OLD' for old data, (no initialization)
= 'NEW' for new data, (initializes all connections)
ncblk = number of s plane connection blocks
ndblk = number of z plane connection blocks
nshblk = number of sample-hold connection blocks

Example 1: A continuous-discrete system with 7 s plane transfer function block, 4 z plane transfer function blocks, and 2 sample-hold blocks is to be modeled for automated transfer function analysis. The model is to be labeled as 'PROJECT ABC'. The FORTRAN code for initializing the connection of this model can be written as:

```
 CALL B2INIT('PROJECT ABC','NEW',7,4,2)
```

PRECMP DIRECTIVE *B2INIT tlabel' oldnew ncblk ndblk nshblk

produces the following FORTRAN statement

```
 CALL B1INIT(
+tlabel
+,oldnew,ncblk,ndblk,nshblk)
```

Example 2: Using Example 1 from above, the following code fragment

```
*B2INIT 'PROJECT ABC' 'NEW' 7 4 2
```

produces the following FORTRAN statement

```
 CALL B2INIT('PROJECT ABC'
+,'NEW',7,4,2)
```

COMMENTS See Chapter 7.

PURPOSE: Loads a file of block diagram connection data previously saved with the command B2SAVE

FORTRAN CALL CALL B2LOAD(file_name)

where, file_name = local file name (.LE.7 characters)
(the local file must be attached before execution of a batch job)

Example 1: Load transfer function connection data for modeling a continuous-discrete multirate system from local file B2DATA. The FORTRAN code can be written as:

```
CALL B2LOAD('B2DATA')
```

PRECMP DIRECTIVE *B2LOAD file_name

produces the following FORTRAN statement

```
CALL B2LOAD(file_name)
```

Example 2:

```
*B2LOAD 'B2DATA'
```

produces the following FORTRAN statement

```
CALL B2LOAD('B2DATA')
```

COMMENTS See the Reference for command B1SAVE for description of the file.

PURPOSE: Automated z plane root loci of a continuous-discrete multirate system modeled as a connection of s and z plane transfer functions and sample-hold blocks

A z plane root locus is automatically computed for a continuous-discrete multirate system modeled as a connection[1] of s and z plane transfer functions and sample-hold blocks. This is a more general root loci command than ZLOCI which computes the root locus by varying the loop gain of a $\mathbf{ZPTF}_i$ transfer function. With the B2LOCI command, the root locus is computed by varying the gain of any $\mathbf{C}_i$ or $\mathbf{D}_i$ block. The gains to be used are multipliers of the transfer function associated with the $\mathbf{C}_i$ or $\mathbf{D}_i$ block selected. Thus, the gains should begin at a value less than 1.0 and end with a value greater than 1.0 if the loci is to bracket the nominal operating gain of the system. The method of specifying the gains to be used in computing the root loci is identical to that used by command ZLOCI. The printout of the state space matrices computed by this command is specified by the parameter PRNMTRX in common block DBASE described by the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
PRNMTRX	0	.EQ.0 No printout of state space matrices .EQ.1 Printout of computed state space matrices .EQ.2 Above plus intermediate matrices

Up to 25 preselected gains for evaluating the root loci can be specified with array KGAIN. The number of preselected gains is determined by NLOCI. The beginning and last gains are determined by KGAIN(1) and KGAIN(NLOCI), respectively. Additional gains can be automatically computed by the program to fill in values between the preselected gains. Between KGAIN(i) and KGAIN(i+1), additional gains will be selected by either of the following two methods if they are between KGAIN(i) and KGAIN(i+1).

If KFLG.EQ.0, gain = KGAIN(i) + KDELT*j, j=1,2, ...

If KFLG.NE.0, gain = KGAIN(i)*KDELT**j, j=1,2, ...

For example, if KGAIN(2)=10, KGAIN(3)=100, KFLG=1, and KDELT=2, the following gains will be used to compute the root loci:

. . . 10., 20., 40., 80., 100., . . .

The total number of gains used is limited by the value of the parameter ITLOC which is preset to 50. Parameters used by B2LOCI are given in the following table.

[1] Specified with calls to subroutines B2INIT, B2CEQ, B2DEQ, B2SEQ, and B2END.

COMMON/DBASE/ parameters used		
parameter	preset	description
NLOCI	2	Number of root locus gains entered in array KGAIN (max=25)
KGAIN	 .5 2.	Array of root locus gains KGAIN(1) = first user-specified root locus gain KGAIN(2) = second user-specified root locus gain KGAIN(NLOCI) = last user-specified root locus gain. (Gains computed and used only if they are between KGAIN(1) and KGAIN(NLOCI))
KFLG	1	.EQ.0 to increment gain by multiplying by KDELT .NE.0 to increment gain by adding by KDELT
KDELT	1.E4	Value for changing gains (preset to large value so that no additional gains are computed).
ITLOC	50	Max. number of different gains computed
GRAFP	1	.NE.0 for low resolution printer plots
HRDCPY	0	.NE.0 for high resolution electrostatic plots
RLXMIN	0.	Min. x axis for plotting
RLXMAX	0.	Max. x axis for plotting (Auto. scaling of x axis if RLXMIN=RLXMAX)
RLYMIN	0.	Min. y axis for plotting
RLYMAX	0.	Max. y axis for plotting (Auto. scaling of y axis if RLYMIN=RLYMAX)
RLFLG1	1	Flag for numbering root locus points on hardcopy plots .EQ.1 for numbering; .EQ.-1 for no numbering

FORTRAN CALL `CALL B2LOCI( blktyp, indx )`

where, blktyp = 'C' if root locus gain is to be applied to a continuous block
'D' if root locus gain is to be applied to a discrete block
indx = block identifier of $\mathbf{C}_i$ or $\mathbf{D}_i$ block where gain is to be varied
(.I.E. number of continuous or discrete connection blocks specified in call to B2INIT)

Example 1: Compute the root locus of continuous-discrete system modeled as a connection of transfer function blocks by varying the gain of block $\mathbf{D}_4$ from 2 to 20 by doubling the first gain until the last gain is reached. Select printer and hardcopy plots with auto scaling. The FORTRAN code can be written as:

```
NLOCI=2
KGAIN(1)=2
KGAIN(2)=20
KFLG=0
KDELT=2
GRAFP=1
HRDCPY=1
CALL B2LOCI('D',4)
```

PRECMP DIRECTIVE *B2LOCI blktyp indx

produces the following FORTRAN statement

```
CALL B2LOCI(blktyp,indx)
```

Example 2:

```
*B2LOCI 'D' 4
```

produces the following FORTRAN statement

```
CALL B2LOCI('D',4)
```

METHOD The Kalman-Bertram state space method is used. See Chapter 7 and Section C.2.

COMMENTS The connection of the transfer function blocks must represent the closed loop configuration of the system.

B2SAVE

PURPOSE: Save the transfer function connection data for modeling a continuous-discrete multirate system to a local file

This subroutine will save the transfer function connection data[1] for modeling a continuous-discrete multirate system. The data will be saved to a local file which the user can reuse in the same job by reloading it with the B2LOAD command. If the file is to be used in a subsequent batch job or an interactive session, it must be SAVEd (CRAY) or CATALOGed (CDC) upon exit from LCAP2.

FORTRAN CALL CALL B2SAVE(file_name)

where, file_name = local file name (.LE.7 characters)

Example 1: Store the transfer function connection data for modeling a continuous-discrete multirate system in a local file named B2DATA. The FORTRAN code can be written as:

```
CALL B2SAVE('B2DATA')
```

PRECMP DIRECTIVE *B2SAVE file_name

produces the following FORTRAN statement

```
CALL B2SAVE(file_name)
```

Example 2:

```
*B2SAVE 'B2DATA'
```

produces the following FORTRAN statement

```
CALL B2SAVE('B2DATA')
```

METHOD See the Reference for command B1SAVE for description of the file.

[1] Specified by calls to subroutines B2INIT, B2CEQ, B2DEQ, B2SEQ, and B2END.

PURPOSE: Equation definition for a $\mathbf{S}_i$ sample-hold block used for automated modeling of a continuous-discrete multirate system by connection of transfer function blocks

Subroutine B2SEQ is part of a set of subroutines used for defining the connection of a set of sample-hold blocks for automated transfer function analysis of a continuous-discrete multirate system. (See B2CEQ and B2DEQ for defining the continuous and discrete blocks). This subroutine is to be called for each $\mathbf{S}_i$ block defined. Before calling B2CEQ, B2DEQ, and B2SEQ, the initialization routine B2INIT must first be called. After the last $\mathbf{C}_i$, $\mathbf{D}_i$, and $\mathbf{S}_i$ blocks are defined by a calls to subroutines B2CEQ, B2DEQ, and B2SEQ, respectively, subroutine B2END must be called to complete the connection procedure.

At the present time the implementation of B2SEQ only allows $\mathbf{S}_i$ block to have only one input, either an output from a $\mathbf{D}_i$ block or another $\mathbf{S}_i$ block. If the input is from a $\mathbf{D}_i$ block, their sampling periods must be equal. If not, modeling of this connection must be changed by inserting another sample-hold block between them which has a sampling period equal to that for the discrete block.

FORTRAN CALL

```
CALL B2SEQ( label, indx, ssampt, delay, nydin, nxsin, iydin, ixsin )
```

where, label = label for block $\mathbf{S}_i$ (.LE.60 characters)
indx = block identifier for block $\mathbf{S}_i$ (.LE.NDIMXS)
ssampt = sampling period of block $\mathbf{S}_i$
delay = delay of block $\mathbf{S}_i$, not implemented yet, but must enter a dummy value
nydin = number of discrete blocks connected as inputs to block $\mathbf{S}_i$ (0 or 1)
nxsin = number of sample-hold blocks connected as inputs to block $\mathbf{S}_i$ (0 or 1)
iydin = array containing the identifiers of the discrete blocks connected to $\mathbf{S}_i$. (negative value for sign change)
ixsin = array containing the identifiers of the sample-hold blocks connected to $\mathbf{S}_i$. (negative value for sign change)

The dummy arrays iydin, and ixsin can be any integer array dimensioned at least nydin, and nxsin, respectively, which the user can declare. For convenience, though, arrays IYDIN, and IXSIN in common block IYCDS can be used instead. These arrays are described in the following table.

COMMON/IYCDS/ parameters used		
parameter	preset	description
IYDIN	0	Array containing the identifiers of the discrete blocks connected to S_i IYDIN(1) = id number of 1st discrete block connected to S_i IYDIN(2) = id number of 2nd discrete block connected to S_i ⋮ IYDIN(nydin) = id number of nydin-th discrete block connected to S_i
IXSIN	0	Array containing the identifiers of the sample-hold blocks connected to S_{indx} IXSIN(1) = id number of 1st sample-hold block connected to S_i IXSIN(2) = id number of 2nd sample-hold block connected to S_i ⋮ IXSIN(nxsin) = id number of nxsin-th sample-hold block connected to S_i

Example 1: The sampling period of block S_3 is 2.5 seconds. There is one discrete blocks, D_2, connected to the it. Block S_3 is to be labeled as 'SLOW RATE ZERO-ORDER HOLD'. The FORTRAN code to define block S_3 can be written as:

```
 INDX=3
 SSAMPT=2.5
 DELAY=0
 NYDIN=1
 IYDIN(1)=2
 NXSIN=0
 CALL B2SEQ('SLOW RATE ZERO-ORDER HOLD',INDX,SSAMPT,DELAY
+,NYDIN,NXSIN,IYDIN,IXSIN)
```

PRECMP DIRECTIVE *B2SEQ label indx ssampt delay nydin nxsin

produces the following FORTRAN statement

```
 CALL B2SEQ  (
+label
+,indx,ssampt,delay,nydin,nxsin,IYDIN,IXSIN)
```

Note that IYDIN and IXSIN are not arguments in the *B2SEQ directive.

Example 2: Using Example 1 from above, the following code fragment

```
*IYDIN 2
      SSAMPT=.5
*B2SEQ 'SLOW RATE ZERO-ORDER HOLD' 3 SSAMPT 0. 1 0
```

produces the following FORTRAN statement

```
 IYDIN(1)=2
 CALL B2SEQ(
+'SLOW RATE ZERO-ORDER HOLD'
+,3,SSAMPT,0.,1,0,IYDIN,IXSIN)
```

METHOD The Kalman-Bertram state space method is used. See Chapter 7.

COMMENTS Part of the automated procedure for implementing the Kalman-Bertram method of analysis is computing the LCM sampling period of the system. In specifying the sampling periods of each z transfer function or sample-hold block, be sure to use enough significant digits. Failure to do so can result in an incorrect LCM calculation. For example, if the sampling rates are 1, 1/3, and 2/3, enter the sampling rates as 1., .3333333, and .6666666. If fewer digits were used instead, i.e., .3333 instead of .3333333, the LCM will be incorrectly computed.

See Example 18 in Chapter 9.

B2TF ·

PURPOSE: Automated SISO transfer function between blocks $\mathbf{D}_i$ and $\mathbf{D}_j$ of a continuous-discrete multirate system modeled as a connection of s and z plane transfer functions and sample-hold blocks
$\mathbf{ZPTF}_i$ = SISO transfer function evaluated

An SISO transfer function is automatically evaluated for a continuous-discrete multirate system modeled as a connection[1] of s and z plane transfer functions and sample-hold blocks. The discrete connection block $\mathbf{D}_i$ is specified by parameter UDIN. The magnitude of the input is specified by the parameter UMAGN. The discrete connection block $\mathbf{D}_j$ which defines the output is specified by the parameter YDOUT. The sampling period of the z plane transfer function computed will be the LCM sampling period.[2] Parameter PRNMTRX is used to specify the type of output for the state space matrices computed by this command. These parameters are in the common block DBASE described in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
UDIN	0	$\mathbf{D}_i$ block number where input u is connected to
UMAGN	1.	Magnitude of the input u
YDOUT	0	$\mathbf{D}_i$ block number defining the output
PRNMTRX	0	.EQ.0 No printout of state space matrices .EQ.1 Printout of computed state space matrices .EQ.2 Above plus intermediate matrices

FORTRAN CALL $\boxed{\text{CALL B2TF(i)}}$

where, i = index for storing evaluated transfer function $\mathbf{ZPTF}_i$

Example 1: Compute the transfer function from $\mathbf{D}_3$ to $\mathbf{D}_6$ for a unit input. Store the resulting transfer function in $\mathbf{ZPTF}_2$. The FORTRAN code can be written as:

```
UDIN=3
YDOUT=6
UMAGN=1.
CALL B2TF(2)
```

PRECMP DIRECTIVE $\boxed{\text{*B2TF i}}$

produces the following FORTRAN statement

```
CALL B1TF(i)
```

[1] Specified by calls to subroutines B2INIT, B2CEQ, B2DEQ, B2SEQ, and B2END.

[2] Lowest-common-multiple of the sampling period of the D_i and S_i blocks (which is computed by the program from data entered in the calls to B2DEQ and B2SEQ).

Example 2:

```
*B2TF 2
```

produces the following FORTRAN statement

```
 CALL B2TF(2)
```

METHOD The Kalman-Bertram state space method is used to model the system. The QR and QZ methods are used to find, respectively, the poles and zeros of the transfer function. See Chapter 7 and Section C.2.

RESTRICTIONS A transfer function can only be computed between two $\mathbf{D}_i$ blocks which are at the LCM sampling period. It is possible to have an LCM sampling period which will be larger than the sampling period of the slowest sampler. For example, a two rate sampled system with sampling periods of 2 and 3 seconds, respectively, will have an LCM sampling period of 6 seconds.

COMMENTS See Example 18 in Chapter 9.

PURPOSE: Compute SISO time response between blocks $\mathbf{D}_i$ and $\mathbf{D}_j$ of a continuous-discrete multirate system modeled as a connection of s and z plane transfer functions and sample-hold blocks

An SISO time response is automatically evaluated for a continuous-disc.rete multirate system modeled as a connection[1] of s and z plane transfer functions and sample-hold blocks. The discrete connection block $\mathbf{D}_i$ which the input is connected to is specified by parameter UDIN. The type of input and its magnitude is specified, respectively, by parameters TTYPE and UMAGN. The discrete connection block $\mathbf{D}_j$ which defines the output is specified by parameter YDOUT. The response will be evaluated at multiples of the LCM sampling period[2] from t=0 to t=TEND. Parameter PRNMTRX is used to specify the type of output for the state space matrices computed by this command. Parameters used by B2TIME are in common block DBASE described in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
UDIN	0	$\mathbf{D}_i$ block number where input u is connected to
TTYPE	1	.EQ.0 for impulse response; .EQ.1 for step response
UMAGN	1.	Magnitude of input
YDOUT	0	$\mathbf{D}_i$ block number defining the output
TEND	1.	End time for evaluating time response
PRNMTRX	0	.EQ.0 No printout of state space matrices .EQ.1 Printout of computed state space matrices .EQ.2 More printout of computed state space matrices
GRAFP	1	.NE.0 for low resolution printer plots
HRDCPY	0	.NE.0 for high resolution electrostatic plots
TXMIN	0.	Minimum x axis for time plot
TXMAX	0.	Maximum x axis for time plot (Auto scaling of x axis if TXMIN=TXMAX)
TYMIN	0.	Minimum y axis for time plot
TYMAX	0.	Maximum y axis for time plot (Auto scaling of y axis if TYMIN=TYMAX)
CONTP	0	=0 Single curve plot =1 First curve of a plot =2 Continuation of a plot =3 Final curve of a plot

FORTRAN CALL CALL B2TIME( )

[1] Specified by calls to subroutines B2INIT, B2CEQ, B2DEQ, B2SEQ, and B2END.

[2] Lowest-common-multiple of the sampling periods of the $\mathbf{D}_i$ and $\mathbf{S}_i$ blocks (which is computed by the program from the data entered in the calls to B2DEQ and B2SEQ).

Example 1: Assume that a system has been modeled as a connection of $\mathbf{SPTF}_i$ and $\mathbf{ZPTF}_i$ transfer functions with calls to subroutines B2INIT, B2CEQ, B2DEQ, B2SEQ, and B2END. Compute the step response from block $\mathbf{D}_3$ to block $\mathbf{D}_7$ from t=0 to t=5 seconds. Select only a high resolution plot.

The FORTRAN code can be written as:

```
UDIN=3
TTYPE=1
UMAGN=1.
YDOUT=7
TEND=5.
GRAFP=0
HRDCPY=1
CALL B2TIME( )
```

PRECMP DIRECTIVE *B2TIME

produces the following FORTRAN statements

```
CALL B2TIME( )
```

METHOD See Section 7.2.2

RESTRICTIONS The sampling periods for both of the discrete connection blocks used to define the input and output of the time response must be equal to the computed LCM sampling period. (Note: The sampling period SAMPT is not used by this command.)

COMMENTS Capability exists for creating multiple plots on the high resolution electrostatic plotter by setting the appropriate value for the parameter CONTP. See Section E.3 for the details.

PURPOSE: Copy polynomials into an s plane transfer function
$\mathbf{SPTF}_i = \mathbf{POLY}_j \;/\; \mathbf{POLY}_k$

FORTRAN CALL CALL CPYPS(i , j , k)

where, i = index of s plane transfer function
j = index of polynomial for the numerator
k = index of polynomial for the denominator

Example 1: Copy $\mathbf{POLY}_2$ and $\mathbf{POLY}_4$ into the numerator and denominator of $\mathbf{SPTF}_5$. The FORTRAN code can be written as:

```
CALL CPYPS(5,2,4)
```

PRECMP DIRECTIVE *CPYPS i j k

produces the following FORTRAN statement

```
CALL CPYPS(i,j,k)
```

Example 2:

```
*CPYPS 5 2 4
```

produces the following FORTRAN statement

```
CALL CPYPS(5,2,4)
```

METHOD Copies coefficients of polynomials into a transfer function. If the roots of the polynomials are also defined, they are copied into the transfer function.

COMMENTS CPYPS is used with commands DTERM or DETRM for evaluating a transfer function from a set of Laplace transformed differential equations.

CPYPW

CPYPW

PURPOSE: Copy polynomials into a w plane transfer function
$\mathbf{WPTF_i = POLY_j / POLY_k}$

This command is similar to CPYPS except that it is for use with a w plane transfer function instead of an s plane transfer function.

CPYPZ

CPYPZ

PURPOSE: Copy polynomials into a z plane transfer function
$\mathbf{ZPTF_i = POLY_j / POLY_k}$

This command is similar to CPYPS except that it is for use with a z plane transfer function instead of an s plane transfer function.

PURPOSE: Copy s plane transfer function into two polynomials
$\mathbf{POLY}_j = \mathbf{SPTF}_i$
$\mathbf{POLY}_k = \mathbf{SPTF}_i$

FORTRAN CALL `CALL CPYSP( i , j , k )`

where, i = index of s plane transfer function
j = index of polynomial for the numerator of the transfer function
k = index of polynomial for the denominator of the transfer function

Example 1: Copy numerator of $\mathbf{SPTF}_5$ into $\mathbf{POLY}_2$ and denominator of $\mathbf{SPTF}_5$ into $\mathbf{POLY}_4$. The FORTRAN code can be written as:

```
CALL CPYSP(5,2,4)
```

PRECMP DIRECTIVE `*CPYSP i j k`

produces the following FORTRAN statement

```
CALL CPYSP(i,j,k)
```

Example 2:

```
*SZXFM 5 2 4
```

produces the following FORTRAN statement

```
CALL CPYSP(5,2,4)
```

METHOD Copies coefficients of a transfer function into two polynomials. If the roots of the transfer function are also defined, they are copied into the polynomials.

CPYWP

CPYWP

PURPOSE: Copy w plane transfer function into two polynomials

$$\mathbf{POLY}_j = \mathbf{WPTF}_i$$

$$\mathbf{POLY}_k = \mathbf{WPTF}_i$$

This command is similar to CPYSP except that it is for use with a w plane transfer function instead of an s plane transfer function.

CPYZP

CPYZP

PURPOSE: Copy z plane transfer function into two polynomials

$$\mathbf{POLY}_j = \mathbf{ZPTF}_i$$

$$\mathbf{POLY}_k = \mathbf{ZPTF}_i$$

This command is similar to CPYSP except that it is for use with a z plane transfer function instead of an s plane transfer function.

PURPOSE: Compute determinant of a matrix with polynomial elements (old version)
POLY$_i$ = determinant of a matrix with polynomial elements

Determinant of a matrix polynomial is used in evaluating an SISO transfer function from a set of Laplace transformed differential equations. Given a system described by:

$$\mathbf{M}(s)\,\mathbf{X}(s) = \mathbf{B}(s)\,u(s)$$

where, $\mathbf{M}(s) = M4\ s^4 + M3\ s^3 + M2\ s^2 + M1\ s + M0$
$\mathbf{B}(s) = B4\ s^4 + B3\ s^3 + B2\ s^2 + B1\ s + B0$
X(s) is a state vector of length MXM
M0, M1, M2, M3, M4 ar› square matrices of order MXM
u is a scalar

by Cramer's method, the transfer function from u to x_j is given by:

$$\frac{x_j(s)}{u(s)} = \frac{\det \mathbf{M}_j(s)}{\det \mathbf{M}(s)}$$

where $\mathbf{M}_j(s)$ is equal to **M**(s) with column j replaced by **B**(s).

This old version of the command (see DTERM for new version) for computing the determinant of a matrix does not have the capability for automatically substituting an input column vector to define the matrix $\mathbf{M}_j(s)$. The user must enter and restore each element of the matrix **M**(s) to form the desired $\mathbf{M}_j(s)$ for each determinant operation.

The parameters used by DETRM are in common block MATRIX1 described in the following table.

COMMON/MATRIX1/ parameters used		
parameter	preset	description
MXM	1	Order of matrices (1-MXMAT)
MDEG	0	Highest degree of polynomial element (0-4)
M0	0	Matrix for coefficients of s^0
M1	0	Matrix for coefficients of s^1
M2	0	Matrix for coefficients of s^2
M3	0	Matrix for coefficients of s^3
M4	0	Matrix for coefficients of s^4

MXMAT = 30 for regular version of LCAP2, MXMAT = 50 for large model version on the CRAY

FORTRAN CALL CALL DETRM(i)

where, i = index of polynomial **POLY**$_i$ for storing the determinant

Example 1: Evaluate the SISO transfer function of a set of Laplace transformed differential equations described by the matrix M(s) and the input vector **B**(s) from input u to output x_3 and store into **SPTF**$_5$. Arbitrarily use **POLY**$_1$ and **POLY**$_2$ to temporarily store the numerator and denominator of the desired transfer function, then copy them into **SPTF**$_5$. The FORTRAN code can be written as:

```
load in data for M(s)
      ⋮
CALL DETRM(1)

      ⋮
enter input vector into column 3 of M(s)
      ⋮
CALL DETRM(2)

      ⋮
restore column 3 of M(s)
      ⋮
CALL CPYPS(5,1,2)
```

PRECMP DIRECTIVE *DETRM i { mxm mdeg }

produces the following FORTRAN statements

```
MXM=mxm        "if 1st optional argument is used"
MDEG=mdeg      "if 2nd optional argument is used"
CALL DETRM(i)
```

Example 2: `*DETRM 3`

produces the following FORTRAN statement

```
CALL DETRM(3)
```

METHOD The determinant is found by solving for its roots directly and then computing its coefficients. See Section C.1.2.

Since this old version of the polynomial determinant command does not automatically insert the input column vector into an appropriate vector of M(s), extra care must be used to ensure that the columns of M(s) are properly restored after each call to DETRM.

RESTRICTIONS The order of the matrices must not be greater than MXMAT and the computed determinant must not be greater than MXPDEG. (MXMAT = 30 and MXPDEG = 49 for normal version of LCAP2, MXMAT = 50 and MXPDEG = 100 for large model version of LCAP2).

COMMENTS For new analysis, this old version of the polynomial determinant command should no longer be used. DETRM is still supported for compatibility with previous versions of LCAP2.

PURPOSE: Compute determinant of a matrix with polynomial elements (new version)
POLY$_i$ = determinant of a matrix with polynomial elements

Determinant of a matrix polynomial is used in evaluating an SISO transfer function from a set of Laplace transformed differential equations. Given a system described by:

$$\mathbf{M}(s)\,\mathbf{y}(s) = \mathbf{B}(s)\,u(s)$$

where, $\mathbf{M}(s) = M4\ s^4 + M3\ s^3 + M2\ s^2 + M1\ s + M0$
$\mathbf{B}(s) = B4\ s^4 + B3\ s^3 + B2\ s^2 + B1\ s + B0$
y(s) is a vector of length MXM
M0, M1, M2, M3, M4 are square matrices of order MXM
B0, B1, B2, B3, B4 are column vectors of length MXM
u is a scalar

by Cramer's method, the transfer function from u to y_j is given by:

$$\frac{y_j(s)}{u(s)} = \frac{\det \mathbf{M}_j(s)}{\det \mathbf{M}(s)}$$

where $\mathbf{M}_j(s)$ is equal to **M**(s) with column j replaced by **B**(s). By definition, $\mathbf{M}_0(s) = \mathbf{M}(s)$.

The parameters used by DTERM are in common block MATRIX1 described in the following table.

COMMON/MATRIX1/ parameters used		
parameter	preset	description
MXM	1	Order of matrices and length of vectors (1-MXMAT)
MDEG	0	Highest degree of polynomial element (0-4)
M0	0	Matrix for coefficients of s^0
M1	0	Matrix for coefficients of s^1
M2	0	Matrix for coefficients of s^2
M3	0	Matrix for coefficients of s^3
M4	0	Matrix for coefficients of s^4
B0	0	Input vector for coefficients of s^0
B1	0	Input vector for coefficients of s^1
B2	0	Input vector for coefficients of s^2
B3	0	Input vector for coefficients of s^3
B4	0	Input vector for coefficients of s^4

MXMAT = 30 for regular version of LCAP2, MXMAT = 50 for large model version on the CRAY

FORTRAN CALL CALL DTERM(i , j)

where, i = index of polynomial $\mathbf{POLY}_i$ for storing the determinant
j = column number for substituting the **B**(s) vector
(0 is defined as no substitution)

Example 1: After loading in a matrix **M**(s) and a vector **B**(s) using arrays M0, ..., M4 and vectors B0, ..., B4, evaluate the SISO transfer function between u and x_3 and store into $\mathbf{SPTF}_5$. Arbitrarily use $\mathbf{POLY}_1$ and $\mathbf{POLY}_2$ to temporarily store the numerator and denominator of the desired transfer function, then copy them into $\mathbf{SPTF}_5$. The FORTRAN code can be written as:

```
    ⋮
CALL DTERM(1,3)
CALL DTERM(2,0)
CALL CPYPS(5,1,2)
```

PRECMP DIRECTIVE *DTERM i j { mxm mdeg }

produces the following FORTRAN statements

```
MXM=mxm       "if 1st optional argument is used"
MDEG=mdeg     "if 2nd optional argument is used"
CALL DTERM(i,j)
```

Example 2: `*DTERM 3 1 2`

produces the following FORTRAN statements

```
MXM=2
CALL DTERM(3,1)
```

METHOD The determinant is found by solving for its roots directly and then computing its coefficients. Although **B**(s) is inserted into column j of **M**(s) for computing the determinant of $\mathbf{M}_j(s)$, the matrices M0, M1, M2, M3, and M4 are not affected. See Section C.1.2.

RESTRICTIONS The order of the matrices must not be greater than MXMAT and the computed determinant must not be greater than MXPDEG. (MXMAT = 30 and MXPDEG = 49 for normal version of LCAP2, MXMAT = 50 and MXPDEG = 100 for large model version of LCAP2).

PURPOSE: Compute frequency response of an arbitrary s plane transfer function using a user supplied subroutine

This command is to be used to compute the frequency response of an s plane transfer function which cannot be represented by an $\mathbf{SPTF}_i$ transfer function (i.e., a nonrational function, a transfer function whose order exceeds MXPDEG, or a transfer function represented as a ratio of polynomial determinants). FREQS is a generalized frequency response command which requires a user-supplied COMPLEX FUNCTION defining the transfer function.

The frequency and plot parameters used by FREQS are in common block DBASE. They are the same ones used by command SFREQ. (See reference on SFREQ)

FORTRAN CALL

```
CALL FREQS(FAUX1)
```

where, FAUX1 is the name of a user supplied subroutine. FAUX1 must be declared with an EXTERNAL statement in the calling routine.

PRECMP DIRECTIVE none

COMMENTS See Example 16 in Chapter 9.

PURPOSE: Compute frequency response of an arbitrary w plane transfer function using a user supplied subroutine

This command is to be used to compute the frequency response of a w plane transfer function which cannot be represented by an **WPTF**$_i$ transfer function (i.e., a nonrational function, a transfer function whose order exceeds MXPDEG, a transfer function which is a function of several different sampling rates, or a hybrid s and w transfer function). FREQW is a generalized frequency response command which requires a user-supplied COMPLEX FUNCTION defining the transfer function.

The frequency and plot parameters used by FREQW are in common block DBASE. They are the same ones used by command WFREQ. (See reference on WFREQ)

FORTRAN CALL

```
CALL FREQW(FAUX1)
```

where, FAUX1 is the name of a user supplied subroutine. FAUX1 must be declared with an EXTERNAL statement in the calling routine.

PRECMP DIRECTIVE none

PURPOSE: Compute frequency response of an arbitrary z plane transfer function using a user supplied subroutine

This command is to be used to compute the frequency response of a z plane transfer function which cannot be represented by an **ZPTF**$_i$ transfer function (i.e., a nonrational function, a transfer function which is a function of several different sampling rates, or a hybrid s and z transfer function). FREQZ is a generalized frequency response command which requires a user-supplied COMPLEX FUNCTION defining the transfer function.

The parameters used by FREQZ are in common block DBASE. They are the same ones used by command ZFREQ. (See reference on ZFREQ)

FORTRAN CALL

```
CALL FREQZ(FAUX1)
```

where, FAUX1 is the name of a user supplied subroutine. FAUX1 must be declared with an EXTERNAL statement in the calling routine.

PRECMP DIRECTIVE none

LOAD LOAD

PURPOSE: Load polynomials, transfer functions and matrix data for a restart

This command allows data previously saved by the SAVE command in either a batch job or an interactive job to be reloaded.

FORTRAN CALL CALL LOAD(file_name, iprn)

where, file_name is a local file name (.LE.7 characters)
(The local file name must be attached before execution of a batch job)
iprn .NE.0 for printout of restored data

Example 1: Load all polynomial, transfer functions, and matrix data from a local file 'OLDDATA' and print out the data. The FORTRAN code can be written as:

```
CALL LOAD('OLDDATA',1)
```

PRECMP DIRECTIVE *LOAD file_name iprn

produces the following FORTRAN statement

```
CALL LOAD(file_name,iprn)
```

Example 2:

```
*LOAD 'OLDDATA' 1
```

produces the following FORTRAN statement

```
CALL LOAD('OLDDATA',1)
```

PURPOSE: Polynomial add

$\mathbf{POLY}_i = \mathbf{POLY}_j + \mathbf{POLY}_k$

FORTRAN CALL CALL PADD(i , j , k)

Example 1: Add $\mathbf{POLY}_4$ to $\mathbf{POLY}_2$ and store into $\mathbf{POLY}_3$. The FORTRAN code can be written as:

```
CALL PADD(3,4,2)
```

PRECMP DIRECTIVE *PADD i j k

produces the following FORTRAN statement

```
CALL PADD(i,j,k)
```

Example 2:

```
*PADD 3 4 2
```

produces the following FORTRAN statement

```
CALL PADD(3,4,2)
```

METHOD After summing the coefficients of $\mathbf{POLY}_j$ and $\mathbf{POLY}_k$ a test is made to determine if there are negligible high order coefficients. If there are, they will be set to zero and the order of the polynomial reduced. The criteria for determining a negligible coefficient is EPAD1 in common block DBASE described in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
EPAD1	1.E-10	Criteria for determining a negligible high order coefficient after a polynomial addition

PURPOSE: Polynomial delete
Deletes polynomial $\mathbf{POLY}_i$ from the current list of polynomial arrays in use

FORTRAN CALL CALL PDEL(i)

Example: Delete $\mathbf{POLY}_3$ from the current list of polynomial arrays in use. The FORTRAN code can be written as:

```
CALL PDEL(3)
```

PRECMP DIRECTIVE *PDEL i

produces the following FORTRAN statement

```
CALL PDEL(i)
```

COMMENTS When the command SAVE is invoked all polynomials, transfer functions, and matrix data will be saved to a file. At the present time there is no means to select only a subset of this data to be saved. The command PDEL, however, allows the user to delete polynomials from the current list of polynomials in use.

PURPOSE: Polynomial equal

$\mathbf{POLY}_i = \mathbf{POLY}_j$

FORTRAN CALL $\boxed{\text{CALL PEQU(i , j)}}$

Example 1: Equate $\mathbf{POLY}_4$ to $\mathbf{POLY}_2$. The FORTRAN code can be written as:

```
CALL PEQU(4,2)
```

PRECMP DIRECTIVE $\boxed{\text{*PEQU i j}}$

produces the following FORTRAN statement

```
CALL PEQU(i,j)
```

Example 2:

```
*PEQU 3 4
```

produces the following FORTRAN statement

```
CALL PEQU(3,4)
```

PLDC

PURPOSE: Load polynomial coefficients into $\mathbf{POLY}_i$
$\mathbf{POLY}_i$ = POLY

To load coefficients into $\mathbf{POLY}_i$, coefficients must first be loaded into the array POLYP. The command PLDC copies the coefficients in POLYP into $\mathbf{POLY}_i$. The array POLYP is in common block POLYCM described in the following table.

COMMON/POLYCM/ parameters used		
parameter	preset	description
POLYP	0.	Array for entering coefficients of a polynomial for use with command PLDC. The format is: POLYP(1) = degree (.LE. MXPDEG) POLYP(2) = coefficient of **0 POLYP(3) = coefficient of **1 . POLYP(n+1) = coefficient of **n

FORTRAN CALL `CALL PLDC( i )`

Example 1: Load the polynomial $s^2 + 75s + 20$ into $\mathbf{POLY}_4$. The first step is to load the coefficients into the array POLYP. The FORTRAN code can be written as:

```
POLYP(1)=2
POLYP(2)=20.
POLYP(3)=75.
POLYP(4)=1.
CALL PLDC(4)
```

PRECMP DIRECTIVE `*PLDC i`

produces the following FORTRAN statement

```
CALL PLDC(i)
```

Example 2:

```
*PLDC 4
```

produces the following FORTRAN statement

```
CALL PLDC(4)
```

RESTRICTIONS The degree of the polynomial must not be greater than MXPDEG. (MXPDEG = 49 for the regular version of LCAP2 and = 100 on the large model version on the CRAY).

PURPOSE: Load polynomial roots into $\mathbf{POLY}_i$
$\mathbf{POLY}_i$ = ROOT

To load roots into $\mathbf{POLY}_i$, roots must first be loaded into the complex array ROOTP. The command PLDR copies the roots in ROOTP into $\mathbf{POLY}_i$. The complex array ROOTP is in common block ROOTCM described in the following table.

COMMON/ROOTCM/ parameters used		
parameter	preset	description
ROOTP	0.	Complex array for entering coefficients of a polynomial for use with command PLDR. The format is: ROOTP(1) = CMPLX(degree,gain) where gain = low order nonzero coefficient ROOTP(2) = root number 1 ROOTP(3) = root number 2 . ROOTP(n+1) = root number n

FORTRAN CALL CALL PLDR(i)

Example 1: Load the polynomial

$$(GAINA)(s+1)(\frac{s}{2-j3}+1)(\frac{s}{2+j3}+1)$$

into $\mathbf{POLY}_4$. The FORTRAN code can be written as:

```
ROOTP(1)=CMPLX(3.,GAINA)
ROOTP(2)=(-1.,0.)
ROOTP(3)=(-2.,3.)
ROOTP(4)=(-2.,-3.)
CALL PLDR(4)
```

PRECMP DIRECTIVE *PLDR i

produces the following FORTRAN statement

```
CALL PLDR(i)
```

Example 2: Using Example 1 from above, the following PRECMP directives

```
*ROOTP GAINA 1. (-2.,-3.)
*PLDR 4
```

will yield a set of FORTRAN statements which will be functionally equivalent to those given in Example 1 above. The FORTRAN code generated by the *ROOTP PRECMP directive includes calls to several subroutines which will convert various root formats (i.e. real, (real,imag), [zeta,omega], and <tau>) into LCAP2 root array format. Note that the number of roots do not have to be entered. It will be computed by the *ROOTP directive. Also, it can optionally allow the user to specify the high order coefficient instead of the low order nonzero coefficient for the "gain". See Reference on *ROOTP in Section 4.2.4 and Example 4.6 in Chapter 4.

RESTRICTIONS The degree of the polynomial must not be greater than MXPDEG. (MXPDEG = 49 for the regular version of LCAP2 and = 100 on the large model version on the CRAY).

PURPOSE: Polynomial multiply
$\mathbf{POLY}_i = \mathbf{POLY}_j * \mathbf{POLY}_k$

FORTRAN CALL CALL PMPY(i , j , k)

Example 1: Multiply $\mathbf{POLY}_4$ by $\mathbf{POLY}_2$ and store into $\mathbf{POLY}_3$. The FORTRAN code can be written as:

```
CALL PMPY(3,4,2)
```

PRECMP DIRECTIVE *PMPY i j k

produces the following FORTRAN statement

```
CALL PMPY(i,j,k)
```

Example 2:

```
*PMPY 3 4 2
```

produces the following FORTRAN statement

```
CALL PMPY(3,4,2)
```

METHOD If the roots of $\mathbf{POLY}_j$ and $\mathbf{POLY}_k$ are available (loaded in or computed in a previous command) the product is computed from the roots instead of from the coefficients.

PURPOSE: Print out polynomial $\mathbf{POLY}_i$

FORTRAN CALL $\boxed{\text{CALL POLY(i)}}$

Example 1: Print out contents of $\mathbf{POLY}_1$. The FORTRAN code can be written as:

```
CALL POLY(1)
```

PRECMP DIRECTIVE $\boxed{\text{*POLY i}}$

produces the following FORTRAN statement

```
CALL POLY(i)
```

Example 2:

```
*POLY 1
```

produces the following FORTRAN statement

```
CALL POLY(1)
```

COMMENTS This command was called PPRN in previous versions of LCAP2.

PURPOSE: Compute roots of polynomial $\mathbf{POLY}_i$

FORTRAN CALL CALL PRTS(i)

Example 1: Compute the roots of $\mathbf{POLY}_2$. The FORTRAN code can be written as:

```
CALL PRTS(2)
```

PRECMP DIRECTIVE *PRTS i

produces the following FORTRAN statement

```
CALL PRTS(i)
```

Example 2: `*PRTS 2`

produces the following FORTRAN statement

```
CALL PRTS(2)
```

METHODS The Aerospace root finding routine MULE [8] is used to find the roots. See Section C.1.

PURPOSE: Polynomial subtract
$\mathbf{POLY}_i = \mathbf{POLY}_j - \mathbf{POLY}_k$

FORTRAN CALL CALL PSUB(i , j , k)

Example 1: Subtract $\mathbf{POLY}_4$ from $\mathbf{POLY}_2$ and store into $\mathbf{POLY}_3$. The FORTRAN code can be written as:

```
CALL PSUB(3,2,4)
```

PRECMP DIRECTIVE *PSUB i j k

produces the following FORTRAN statement

```
CALL PSUB(i,j,k)
```

Example 1: *PSUB 3 2 4

produces the following FORTRAN statement

```
CALL PSUB(3,2,4)
```

METHOD After subtracting the coefficients of $\mathbf{POLY}_k$ from the coefficients of $\mathbf{POLY}_j$, a test is made to determine if there are negligible high order coefficients. If there are, they will be set to zero and the order of the polynomial reduced. The criteria for determining a negligible coefficient is EPAD1 in common block DBASE described in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
EPAD1	1.E-10	Criteria for determining a negligible high order coefficient after a polynomial subtraction

PURPOSE: Save polynomials, transfer functions, and matrix data for a restart

This command will save polynomials, transfer functions, and matrix data (used by DTERM and DETRM) for use in a subsequent LCAP2 batch job or an interactive LCAP2 job. Data in the character parameter SAVLBL in common block HEAD described below is used as a header in the stored file.

COMMON/HEAD/ parameters used		
parameter	preset	description
SAVLBL	' '	character parameter for labeling file used for STORE command

FORTRAN CALL

```
CALL SAVE (file_name, iprn )
```

where, file_name is a local file name (.LE.7 characters)
(The local file must be SAVEd (CRAY) or CATALOGed (CDC) after execution of a batch job)
iprn .NE. 0 for printout of data saved

Example 1: Save all polynomial, transfer functions, and matrix data to file 'NEWDATA' and printout the data. Label the data as 'final data, 4/88'. The required FORTRAN code is:

```
SAVLBL='FINAL DATA, 4/88'
CALL SAVE('NEWDATA',1)
```

PRECMP DIRECTIVE

```
*SAVE file_name iprn
```

produces the following FORTRAN statement

```
CALL SAVE(file_name,iprn)
```

Example 2:

```
*SAVE 'NEWDATA' 1
```

produces the following FORTRAN statement

```
CALL SAVE('NEWDATA',1)
```

COMMENTS All polynomial, transfer function, and matrix data currently in use will be written to a local file. At the present time there is no means to select only a subset of the data to be saved. However, the commands PDEL, SPDEL, ZPDEL, and WPDEL can be used to delete any polynomial or transfer function before invoking the command SAVE.

PURPOSE: Eliminate common roots of s plane transfer function **SPTF**$_i$

FORTRAN CALL CALL SELCR(i)

Example 1: Eliminate common roots of **SPTF**$_2$. The FORTRAN code can be written as:

```
CALL SELCR(2)
```

PRECMP DIRECTIVE *SELCR i

produces the following FORTRAN statement

```
CALL SELCR(i)
```

Example 2:

```
*SELCR 2
```

produces the following FORTRAN statement

```
CALL SELCR(2)
```

METHOD If a numerator root nrt and a denominator root drt are found such that (1) CABS(drt/nrt - (1.,0.)) .LT. ECRE1 for nrt .NE. 0. or (2) CABS(drt). LT. ECRE2 for nrt .EQ. 0., roots nrt and drt are considered to be common and will be eliminated from the transfer function. ECRE1 and ECRE2 are in common block DBASE described in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
ECRE1	2.E-4	Tolerance for eliminating common roots in subroutine CRELIM
ECRE2	1.E-8	Tolerance for "zero" root in subroutine CRELIM

See Section C.8.

PURPOSE: Compute frequency response of s plane transfer function $\mathbf{SPTF}_i$

Two modes are available for selecting the frequency points used for evaluating the frequency response of a transfer function stored in $\mathbf{SPTF}_i$. The automatic mode, selected when FAUTO is nonzero, allows the user to specify up to 20 preselected frequencies with the array OMEGA. In this mode the program will use all these preselected points plus its own dynamically generated points to yield a smooth plot. The number of preselected points is determined by NOMEGA. The beginning and last frequency points for computing the response are determined by OMEGA(1) and OMEGA(NOMEGA), respectively.

In the nonautomatic frequency mode (FAUTO=0) the user can define up to five sets of frequencies to be used in computing the response. Each of these sets is specified by a three element array of the form FREQk(i), i=1,3. If FREQk(1)=a, FREQk(2)=b and FREQk(3)=c, the k-th set of frequencies specified is:

a, a+c, a+2c, ... a+jc, b

where j is the largest integer such that (a+jc) is less than b. Each successive FREQk array must define an increasing set of frequencies such that the first value of the segment is always larger than the last value of the preceding segment. When FREQk(3) is not larger than FREQk(1), as in the case with the preset values for $k = 2, 5$, those segments will not be used.

With either the automatic or the nonautomatic frequency mode, the program will automatically check for the gain and phase crossover. When found, the program will iterate until the exact crossover frequency is found. The limit on the number of plot points computed is 1500.

The units used for representing transfer functions are in rad/sec. For frequency response calculations, the frequencies used for evaluating the response can either be in rad/sec or Hz. A nonzero value for the parameter RAD will select the units rad/sec, while a zero value will select Hz.

There is a provision to include a time delay, e^{-ds}, in cascade with $\mathbf{SPTF}_i$ when computing the frequency response. This time delay is specified by the parameter FDELAY.

Bode, Nichols, and Nyquist plots are selected by the values of the flags FBODE, FNICO, FNYQS, respectively. The more commonly used parameters by SFREQ in common block DBASE are given in the following table. They include plot parameters.

If a Bode plot is selected and the user enters a starting frequency of 0.0 instead of a nonzero value, the program will plot the response over 6 cycles (ignoring the value of CYCLE) with a starting value equal to 1.E-6*(largest frequency value selected).

COMMON/DBASE/ parameters used

parameter	preset	description
FAUTO	1	.NE.0 for automatic frequency point selection. Uses NOMEGA and array OMEGA. .EQ.0 for user supplied frequency points. Uses arrays FREQ1, FREQ2, ..., FREQ5.
NOMEGA	2	Number of preselected frequency points in array OMEGA for use in auto. frequency mode. (range = 2 - 20)
OMEGA		Array of preselected frequency points for auto. frequency mode. (units determined by RAD)
	1.	OMEGA(1) = first frequency point used in auto. mode
	10.	OMEGA(2) = second frequency point used in auto. mode
		OMEGA(NOMEGA) = last frequency point used in auto. mode
RAD	1	.NE.0 for rad/sec, otherwise Hz
FBODE	1	.NE.0 for Bode plot
FNICO	0	.NE.0 for Nichols plot
PMARG	0	.NE.0 for plotting phase margin instead of phase for the Nichols plot
FNYQS	0	.NE.0 for Nyquist plot
NQDB	0	.NE.0 for hardcopy Nyquist plot in dB when used with FNYQS
GRAFP	1	.NE.0 for low resolution printer plots
HRDCPY	0	.NE.0 for high resolution electrostatic plots
FDELAY	0.	Time delay (dead time) for s plane frequency response
DEGMN	-360.	Minimum defined phase in frequency response (Phase defined from DEGMN to DEGMN+360.)
CYCLE	0	Number of log cycles for Bode plot (0-6) .EQ. 0 for automatic selection
FREQ1(1)	1.	Starting freq. point for first segment of user specified values (when FAUTO=0). (units determined by RAD)
FREQ1(2)	10.	End freq. point for first segment of user specified values (when FAUTO=0).
FREQ1(3)	1.	Delta frequency for first segment of user specified values (when FAUTO=0).
FREQk(1)	0	Starting freq. point for k-th segment of user specified values (when FAUTO=0). k=2,5
FREQk(2)	0	End freq. point for k-th segment of user specified values (when FAUTO=0). k=2,5
FREQk(3)	0	Delta frequency for k-th segment of user specified values (when FAUTO=0). k=2,5
DBMAX	0.	Maximum dB for plotting frequency response
DBMIN	0.	Minimum dB for plotting frequency response. (Auto. scaling if DBMIN=DBMAX)
FXYDEL	.5	Nyquist plot scale in units per inch. (Auto scaling if FXYDEL=0)
FXYMIN	-2.5	Nyquist plot parameter - minimum real and imag. value plotted

FORTRAN CALL CALL SFREQ(i)

Example 1: Compute the frequency response of SPTF$_3$, using the auto frequency mode for frequencies between 0. and 1000. and at 1., 10., and 100. Select printer and hardcopy Bode and Nichols plots with auto scaling. The FORTRAN code can be written as:

```
FAUTO=1
NOMEGA=5
OMEGA(1)=0.
OMEGA(2)=1.
OMEGA(3)=10.
OMEGA(4)=100.
OMEGA(5)=1000.
FBODE=1
FNICO=1
GRAFP=1
HRDCPY=1
CALL SFREQ(3)
```

Example 2: Compute the frequency response of SPTF$_3$, using the nonauto frequency mode for frequencies between 0. and 2. in increments of .1; between 3. and 20. in increments of 1.; and between 22. and 100. in increments of 2. Select printer and hardcopy Bode and Nichols plots with fixed scaling between -30 and 10 for the dB axis. The FORTRAN code can be written as:

```
FAUTO=0
FREQ1(1)=0.
FREQ1(2)=2.
FREQ1(3)=.1
FREQ2(1)=3.
FREQ2(2)=20.
FREQ2(3)=1.
FREQ3(1)=22.
FREQ3(2)=100.
FREQ3(3)=2.
FBODE=1
FNICO=1
DBMIN=-30
DBMAX=10
CALL SFREQ(3)
```

PRECMP DIRECTIVE *SFREQ i

produces the following FORTRAN statement

```
CALL SFREQ(i)
```

Example 3: Using Example 1 from above, the following code fragment

```
      FAUTO=1
      NOMEGA=5
*OMEGA 0 1 10 100 1000
      FBODE=1
      FNICO=1
      GRAFP=1
      HRDCPY=1
*SFREQ 3
```

produces the same FORTRAN statements as in Example 1.

METHOD The frequency response can be evaluated by using either (1) the coefficient form or (2) the root form (if available) of **SPTF**$_i$. If the roots are available, that form will be used since the response can be computed with higher accuracy.[1]

If the automatic frequency mode is selected (FAUTO.NE.0), the program will choose frequency points for evaluating the response such that successive dB and phase values will be within specified limits to yield a smooth plot. The program evaluates the first point using f = OMEGA(1). Then choosing deltaf = OMEGA(1)/20 initially, the next frequency to be used is computed as f =f + deltaf. Evaluating the response using this value of f, the delta dB and phase are compared to the specified limits. If either is too large, deltaf is halved and the response is recomputed. If both are too small, deltaf is doubled and the response is recomputed. The limits for delta dB are EDB1/2 and EDB1. The limits for delta phase is EDEG1/2 and EDEG1. Simultaneously with computing the next f be to used in evaluating the response, a comparison is made with the next value of OMEGA(i). If f is larger than OMEGA(i), f will be replaced with the value of OMEGA(i). This will ensure that the user's prespecified frequency points will be used. This procedure will continue until the last value of OMEGA(NOMEGA) is used. There is a limit on the number of iterations in computing the frequency points. This limit can be changed by the parameter MAXITF.

If a frequency point being used to evaluate the response is equal to a pole on the $j\omega$, the warning message

ZERO IN DENOMINATOR AT OMEGA = ___ , THIS POINT SKIPPED

will be printed out and the program allowed to continue. If this situation occurs when the automatic

[1] Also, for very high order transfer functions, the following problems will be avoided: (1) Premature overflow of the numerator or denominator terms if the coefficients are scaled by a very large number. (This can occur if a large number of block diagram reduction operations were used to compute **SPTF**$_i$. For this situation, though, the command SNORM can be used to normalize the coefficients.) and (2) Inherently less accurate representation of the coefficients in a finite machine (the coefficients are functions of terms involving products of the roots) will become pronounced as the order of the transfer function is increased.

frequency mode is selected and OMEGA(1)=0. (i.e., when the transfer function has a pole at the origin), the program will be restarted with the first frequency point at OMEGA(2)/1000[1] instead.

Also as part of the automatic frequency mode, a comparison is made on deltaf to keep deltaf/f within the limits of MINDW and MAXDW. The lower limit MINDW is necessary to prevent an excessive number of plot points around frequencies with very low damping coefficients. The upper limit MAXDW will ensure that there are enough points to yield a smooth Bode plot.

Since the plot points computed to generate a smooth plot will, in many cases, be very large, only a portion of the computed response will be printed out. The printout is controlled by the delta dB and delta phase parameters, EDB2 and EDEG2, respectively. A tabular printout is made only if either of these limits are exceeded.

Capability exists for creating multiple plots on the high resolution electrostatic printer by setting the appropriate value for the parameter CONTP. See Section E.3 for the details.

The following table lists additional parameters in the common block DBASE which the user can change.

COMMON/DBASE/ parameters used		
parameter	preset	description
CONTP	0	=0 Single curve plot =1 First curve of a plot =2 Continuation of a plot =3 Final curve of a plot
EDB1	1.	Min. delta dB for plotting
EDB2	2.	Min. delta dB for printout
EDEG1	4.	Max. delta degree for plotting
EDEG2	10.	Max. delta degree of printout
MINDW	.0005	Min. relative frequency step size
MAXDW	.2	Max. relative frequency step size
MAXITF	3000	Max. number of iterations allowed

[1] This will allow a nonzero deltaf = (OMEGA(2)/1000)/20 to be used instead of OMEGA(1)/20 for initializing the variable frequency step size.

PURPOSE: Compute root loci of s plane transfer function $\mathbf{SPTF}_i$

Root locus of an open loop transfer function $\mathbf{SPTF}_i$ is computed by varying the loop gain. The gains to be used by SLOCI are multipliers of $\mathbf{SPTF}_i$. Thus, the gains should begin at a value less than 1.0 and end at a value greater than 1.0 if the loci is to bracket the nominal operating gain of the system. Up to 25 preselected gains for evaluating the root loci can be specified with array KGAIN. The number of preselected gains is determined by NLOCI. The beginning and last gains are determined by KGAIN(1) and KGAIN(NLOCI), respectively. Additional gains can be automatically computed by the program to fill in values between the preselected gains. Between KGAIN(i) and KGAIN(i+1), additional gains will be selected by either of the following two methods if they are between KGAIN(i) and KGAIN(i+1).

If KFLG.EQ.0, gain = KGAIN(i) + KDELT*j, j=1,2, ...

If KFLG.NE.0, gain = KGAIN(i)*KDELT**j, j=1,2, ...

For example, if KGAIN(2)=10, KGAIN(3)=100, KFLG=1, and KDELT=2, the following gains will be used to compute the root loci:

. . . 10., 20., 40., 80., 100., . . .

The total number of gains used is limited by the value of the parameter ITLOC which is preset to 50. Parameters used by SLOCI are given in the following table.

COMMON/DBASE/ parameters used

parameter	preset	description
NLOCI	2	Number of root locus gains entered in array KGAIN (max=25)
KGAIN		Array of root locus gains
	.5	KGAIN(1) = first user-specified root locus gain
	2.	KGAIN(2) = second user-specified root locus gain
		KGAIN(NLOCI) = last user-specified root locus gain.
		(Gains computed and used only if they are between KGAIN(1) and KGAIN(NLOCI))
KFLG	1	.EQ.0 to increment gain by multiplying by KDELT .NE.0 to increment gain by adding by KDELT
KDELT	1.E4	Value for changing gains (preset to large value so that no additional gains are computed).
ITLOC	50	Max. number of different gains computed
GRAFP	1	.NE.0 for low resolution printer plots
HRDCPY	0	.NE.0 for high resolution electrostatic plots
RLXMIN	0.	Min. x axis for plotting
RLXMAX	0.	Max. x axis for plotting (Auto. scaling if RLXMIN=RLXMAX)
RLYMIN	0.	Min. y axis for plotting
RLYMAX	0.	Max. y axis for plotting (Auto. scaling if RLYMIN=RLYMAX)
RLFLG1	1	Flag for numbering root locus points on hardcopy plots .EQ.1 for numbering; .EQ.-1 for no numbering

FORTRAN CALL CALL SLOCI(i)

Example 1: Compute the root locus of $\mathbf{SPTF}_2$ by varying the gain from .5 to 20 by doubling the first gain until the last gain is reached. Select printer and hardcopy plots with auto scaling. The FORTRAN code can be written as:

```
NLOCI=2
KGAIN(1)=.5
KGAIN(2)=20.
KFLG=0
KDELT=2
GRAFP=1
HRDCPY=1
CALL SLOCI(2)
```

PRECMP DIRECTIVE *SLOCI i

produces the following FORTRAN statement

```
CALL SLOCI(I)
```

Example 2: Using Example 1 from above, the following code fragment

```
      NLOCI=2
*KGAIN .5 20.
      KFLG=0
      KDELT=2
      GRAFP=1
      HRDCPY=1
*SLOCI 2
```

produces the following FORTRAN statements

```
NLOCI=2.
KGAIN(1)=.5
KGAIN(2)=20.
KFLG=0.
KDELT=2.
GRAFP=1.
HRDCPY=1.
CALL SLOCI(2)
```

METHOD Root locus is computed by evaluating the roots of the polynomial (PN + k*PD) where k is the gain to be varied and PN and PD are the numerator and denominator polynomials of the transfer function. For a more general root locus where the gain to be varied is not the loop gain, see command B1LOCI.

SNORM

PURPOSE: Normalize coefficients of s plane transfer function $\mathbf{SPTF}_i$

Normalization can be either with respect to the low order nonzero coefficient or the high order coefficient of the denominator. The parameter NRMHI determines which method is to be used.

If NRMHI is equal to zero, the low order nonzero coefficient of the denominator is set equal to the parameter KNORM and all other coefficients are normalized to this value. If KNORM = 1., the low order nonzero coefficient of the numerator is the Bode gain.

If NRMHI is not equal to zero, the high order coefficient of the denominator is set equal to the parameter KNORM and all other coefficients are normalized to this value. If KNORM = 1., the high order coefficient of the numerator is the root locus gain. Parameters used by SNORM are given in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
KNORM	1.	Value used for normalizing the transfer function
NRMHI	0	.EQ.0 for normalizing to the low order nonzero coefficient .NE.0 for normalizing to the high order coefficient

FORTRAN CALL CALL SNORM(i)

Example 1: If the s plane transfer function

$$\frac{s + 14}{2s^2 + 7s + 78}$$

stored in $\mathbf{SPTF}_1$, is to be normalized so that the low order denominator coefficient is 1.0 (so that the Bode gain appears explicitly as the low order numerator coefficient), the FORTRAN code can be written as:

```
NRMHI=0
KNORM=1.
CALL SNORM(1)
```

PRECMP DIRECTIVE *SNORM i { nrmhi knorm }

produces the following FORTRAN statements

```
NRMHI=nrmhi          "if 1st optional argument is used"
KNORM=knorm          "if 2nd optional argument is used"
CALL SNORM(i)
```

Example: `*SNORM 1 0 2`

produces the following FORTRAN statement

```
NRMHI=0
KNORM=2
CALL SNORM(1)
```

COMMENTS When the block diagram reduction method is used, the resulting transfer function of interest may have its numerator and denominator coefficients scaled by a very large number if the system order is very high. This may lead to an unnecessary overflow in a subsequent frequency response evaluation since either or both the numerator and denominator terms may overflow while the ratio (numerator term/denominator term) is within the range of the computer. Command SNORM can be used to scale the coefficients so that this type of overflow can be prevented.

PURPOSE: S plane transfer function add
$\mathbf{SPTF}_i = \mathbf{SPTF}_j + \mathbf{SPTF}_k$

FORTRAN CALL CALL SPADD(i , j , k)

Example: Add $\mathbf{SPTF}_4$ to $\mathbf{SPTF}_2$ and store into $\mathbf{SPTF}_3$ The FORTRAN code can be written as:

```
CALL SPADD(3,4,2)
```

PRECMP DIRECTIVE *SPADD i j k

produces the following FORTRAN statement

```
CALL SPADD(i,j,k)
```

Example: `*SPADD 3 4 2`

produces the following FORTRAN statement

```
CALL SPADD(3,4,2)
```

COMMENTS If the roots of both $\mathbf{SPTF}_j$ and $\mathbf{SPTF}_k$ are available (loaded in or computed from a previous command) any common roots between the denominators will be factored out before the sum is computed.

PURPOSE: S plane closed loop transfer function
$\mathbf{SPTF}_i = \mathbf{SPTF}_j \,/\, (1 + \mathbf{SPTF}_j * \mathbf{SPTF}_k)$

FORTRAN CALL $\boxed{\text{CALL SPCLSLP(i , j , k)}}$

Example: Compute closed loop transfer function $\mathbf{SPTF_4}$ where $\mathbf{SPTF_2}$ is the forward loop transfer function and $\mathbf{SPTF_3}$ is the feedback transfer function. The FORTRAN code can be written as:

```
CALL SPCLSLP(4,2,3)
```

PRECMP DIRECTIVE $\boxed{\text{*SPCLSLP i j k}}$

produces the following FORTRAN statement

```
CALL SPCLSLP(i,j,k)
```

Example:
```
*SPCLSLP 4 2 3
```

produces the following FORTRAN statement

```
CALL SPCLSLP(4,2,3)
```

PURPOSE: S plane transfer function delete
Deletes transfer function $\mathbf{SPTF}_i$ from the current list of transfer function arrays in use

FORTRAN CALL CALL SPDEL(i)

Example: Delete $\mathbf{SPTF}_3$ from the current list of transfer function arrays in use. The FORTRAN code can be written as:

```
CALL SPDEL(3)
```

PRECMP DIRECTIVE *SPDEL i

produces the following FORTRAN statement

```
CALL SPDEL(i)
```

COMMENTS When the command SAVE is invoked all polynomials, transfer functions, and matrix data will be saved to a file. At the present time there is no means to select only a subset of this data to be saved. The command SPDEL, however, allows the user to delete s plane transfer functions from the current list of transfer functions in use.

PURPOSE: S plane transfer function divide
$\mathbf{SPTF}_i = \mathbf{SPTF}_j \,/\, \mathbf{SPTF}_k$

FORTRAN CALL $\boxed{\text{CALL SPDIV(i , j , k)}}$

Example: Divide $\mathbf{SPTF}_4$ by $\mathbf{SPTF}_2$ and store into $\mathbf{SPTF}_3$. The FORTRAN code can be written as:

```
CALL SPDIV(3,4,2)
```

PRECMP DIRECTIVE $\boxed{\text{*SPDIV i j k}}$

produces the following FORTRAN statement

```
CALL SPDIV(i,j,k)
```

Example: `*SPDIV 3 4 2`

produces the following FORTRAN statement

```
CALL SPDIV(3,4,2)
```

METHOD If the roots of $\mathbf{SPTF}_j$ and $\mathbf{SPTF}_k$ are available (loaded in or computed in a previous command) the quotient is computed from the roots instead of from the coefficients.

SPEQU

PURPOSE: S plane transfer function equal
$\mathbf{SPTF}_i = \mathbf{SPTF}_j$

FORTRAN CALL CALL SPEQU(i , j)

Example: Equate $\mathbf{SPTF}_4$ to $\mathbf{SPTF}_2$. The FORTRAN code can be written as:

```
CALL SPEQU(4,2)
```

PRECMP DIRECTIVE *SPEQU i j

produces the following FORTRAN statement

```
CALL SPEQU(i,j)
```

Example: `*SPEQU 4 2`

produces the following FORTRAN statement

```
CALL SPEQU(4,2)
```

PURPOSE: Load transfer function coefficients into **SPTF**$_i$
SPTF$_i$ = POLYN / POLYD

To load coefficients into **SPTF**$_i$, coefficients must first be loaded into the arrays POLYN and POLYD. The command SPLDC copies the coefficients in POLYN and POLYD into **SPTF**$_i$. The arrays POLYN and POLYD are in common block POLYCM described in the following table:

COMMON/POLYCM/ parameters used		
parameter	preset	description
POLYN	0.	Array for entering coefficients of a numerator polynomial for use with command SPLDC, WPLDC, ZPLDC. The format is: POLYN(1) = degree (.LE. MXPDEG) POLYN(2) = coefficient of **0 POLYN(3) = coefficient of **1 ⋮ POLYN(n+1) = coefficient of **n
POLYD	0.	Array for entering coefficients of a denominator polynomial for use with command SPLDC, WPLDC, ZPLDC. The format is: POLYD(1) = degree (.LE. MXPDEG) POLYD(2) = coefficient of **0 POLYD(3) = coefficient of **1 ⋮ POLYD(n+1) = coefficient of **n

FORTRAN CALL `CALL SPLDC( i )`

Example 1: Load the transfer function

$$\frac{8s+3}{17s^2+6s+11}$$

into **SPTF**$_4$. The first step is to load the numerator and denominator into POLYN and POLYD, respectively. The FORTRAN code can be written as:

```
POLYN(1)=1
POLYN(2)=3
POLYN(3)=8
POLYD(1)=2
POLYD(2)=11
POLYD(3)=6
POLYD(4)=17
CALL SPLDC(4)
```

PRECMP DIRECTIVE $\boxed{\text{*SPLDC i}}$

produces the following FORTRAN statement

```
CALL SPLDC(i)
```

Example 2: Using Example 1 from above, the following PRECMP directives

```
*POLYN 1 3 8
*POLYD 2 11 6 17
*SPLDC 4
```

produces the same FORTRAN code given in Example 1. See Section 4.2.4 for definition of the *POLYN and *POLYD directives.

RESTRICTIONS The degree of the numerator and denominator must be .LE.MXPDEG. (MXPDEG = 49 for the regular version of LCAP2, = 100 for large model version on the CRAY).

PURPOSE: Load transfer function roots into $\mathbf{SPTF}_{i}$
$\mathbf{SPTF}_{i}$ = ROOTN / ROOTD

To load roots into $\mathbf{SPTF}_{i}$, coefficients must first be loaded into the arrays ROOTN and ROOTD. The command SPLDR copies the coefficients in ROOTN and ROOTD into $\mathbf{SPTF}_{i}$. The complex arrays ROOTN and ROOTD are in common block ROOTCM described in the following table:

COMMON/ROOTCM/ parameters used		
parameter	preset	description
ROOTN	0.	Complex array for entering numerator roots into a transfer function for use with commands SPLDC, WPLDC, or ZPLDC. The format is: ROOTN(1) = CMPLX(degree,gainn) where gainn = low order nonzero numerator coefficient ROOTN(2) = numerator root number 1 ROOTN(3) = numerator root number 2 ⋮ ROOTN(n+1) = numerator root number n
ROOTD	0.	Complex array for entering numerator roots into a transfer function for use with commands SPLDC, WPLDC, or ZPLDC. The format is: ROOTD(1) = CMPLX(degree,gaind) where gaind = low order nonzero denominator coefficient ROOTD(2) = denominator root number 1 ROOTD(3) = denominator root number 2 ⋮ ROOTD(n+1) = denominator root number n

FORTRAN CALL `CALL SPLDR( i )`

Example 1: Load the transfer function

$$\frac{50(\frac{s}{2}+1)}{7(\frac{s}{4}+1)(\frac{s}{3-j5}+1)(\frac{s}{3+j5}+1)}$$

into **SPTF**$_2$. The first step is to load the numerator and denominator into the complex arrays ROOTN and ROOTD, respectively. The FORTRAN code can be written as:

```
ROOTN(1)=(1.,50.)
ROOTN(2)=(-2.,0.)
ROOTD(1)=(3.,7.)
ROOTD(2)=(-4.,0.)
ROOTD(3)=(-3.,5.)
ROOTD(4)=(-3.,-5.)
CALL SPLDR(2)
```

PRECMP DIRECTIVE *SPLDR i

produces the following FORTRAN statement

```
CALL SPLDR(i)
```

Example 2: Using Example 1 from above, the following PRECMP directives

```
*ROOTN 50. -2
*ROOTD 7 -4 (-3.,5.)
*SPLDR 2
```

will yield a set of FORTRAN statements which will be functionally equivalent to those given in Example 1 above. The FORTRAN code generated by the *ROOTN and *ROOTD PRECMP directives includes calls to several subroutines which will convert various root formats (i.e. real, (real,imag), [zeta,omega], and <tau>) into LCAP2 root array format. Note that the number of roots does not have to be entered. It will be computed by the *ROOTN and *ROOTD directives. Also, it can optionally allow the user to specify the high order coefficients instead of the low order nonzero coefficients for the numerator and denominator "gains". See Section 4.2.4 for description of *ROOTN and *ROOTD.

RESTRICTIONS The degree of the numerator and denominator must be .LE. MXPDEG. (MXPDEG = 49 for the regular version of LCAP2, = 100 for the large order model on the CRAY).

SPMPY SPMPY

PURPOSE: S plane transfer function multiply
$\mathbf{SPTF}_i = \mathbf{SPTF}_j * \mathbf{SPTF}_k$

FORTRAN CALL CALL SPMPY(i , j , k)

Example: Multiply $\mathbf{SPTF}_4$ and $\mathbf{SPTF}_2$ and store into $\mathbf{SPTF}_3$. The FORTRAN code can be written as:

```
CALL SPMPY(3,4,2)
```

PRECMP DIRECTIVE *SPMPY i j k

produces the following FORTRAN statement

```
CALL SPMPY(i,j,k)
```

Example:

```
*SPMPY 3 4 2
```

produces the following FORTRAN statement

```
CALL SPMPY(3,4,2)
```

METHOD If the roots of $\mathbf{SPTF}_j$ and $\mathbf{SPTF}_k$ are available (loaded in or computed in a previous command) the product is computed from the roots instead of from the coefficients.

SPTF
alias: **SPPRN**

PURPOSE: Print out s plane transfer function $\mathbf{SPTF}_i$

FORTRAN CALL CALL SPTF(i)

Example: Print out contents of $\mathbf{SPTF}_1$. The FORTRAN code can be written as:

```
CALL SPTF(1)
```

PRECMP DIRECTIVE *SPTF i

produces the following FORTRAN statement

```
CALL SPTF(i)
```

Example: `*SPPRN 1`

produces the following FORTRAN statement

```
CALL SPPRN(1)
```

COMMENTS This command was called SPPRN in previous versions of LCAP2.

PURPOSE: Compute roots of s plane transfer function $\mathbf{SPTF}_i$

FORTRAN CALL CALL SPRTS(i)

Example 1: Compute the roots of $\mathbf{SPTF}_2$. The FORTRAN code can be written as:

```
CALL SPRTS(2)
```

PRECMP DIRECTIVE *SPRTS i

produces the following FORTRAN statement

```
CALL SPRTS(i)
```

Example 2:

```
*SPRTS 2
```

produces the following FORTRAN statement

```
CALL SPRTS(2)
```

METHOD The Aerospace root finding MULE [8] routine is used to find the roots. See Section C.1.

PURPOSE: S plane transfer function subtract
$\mathbf{SPTF}_i = \mathbf{SPTF}_j - \mathbf{SPTF}_k$

FORTRAN CALL CALL SPSUB(i . i , k)

Example: Subtract $\mathbf{SPTF}_4$ from $\mathbf{SPTF}_2$ and store into $\mathbf{SPTF}_3$. The FORTRAN code can be written as:

```
CALL SPSUB(3,2,4)
```

PRECMP DIRECTIVE *SPSUB i j k

produces the following FORTRAN statement

```
CALL SPSUB(i,j,k)
```

Example: *SPSUB 3 2 4

produces the following FORTRAN statement

```
CALL SPSUB(3,2,4)
```

COMMENTS If the roots of both $\mathbf{SPTF}_j$ and $\mathbf{SPTF}_k$ are available (loaded in or computed from a previous command) any common roots between the denominators will be factored out before the difference is computed.

PURPOSE: Inverse Laplace transform and time response of s plane transfer function $\mathbf{SPTF}_i$

Compute inverse Laplace transform analytically and evaluate the time response. The input can be a step or input as specified by the parameter TTYPE. The magnitude is specified by the parameter TMAGN. The response will be evaluated from TZERO to TEND in increments of TDELT. Since the response is evaluated analytically, TZERO does not have to be zero. Parameters used by STIME are in common block DBASE described in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
TTYPE	1	.EQ.0 for impulse response; .EQ.1 for step response
TMAGN	1.	Magnitude of input
TZERO	0.	Start time for evaluating time response
TEND	1.	End time for evaluating time response
TDELT	1.	Delta time for evaluating s plane time response
GRAFP	1	.NE.0 for low resolution printer plots
HRDCPY	0	.NE.0 for high resolution electrostatic plots
TXMIN	0.	Minimum x axis for time plot
TXMAX	0.	Maximum x axis for time plot (Auto scaling of x axis if TXMIN=TXMAX)
TYMIN	0.	Minimum y axis for time plot
TYMAX	0.	Maximum y axis for time plot (Auto scaling of y axis if TYMIN=TYMAX)
CONTP	0	=0 Single curve plot =1 First curve of a plot =2 Continuation of a plot =3 Final curve of a plot

FORTRAN CALL CALL STIME(i)

Example 1: Compute the step response of $\mathbf{SPTF}_2$ from 5 to 25 seconds in increments of 0.5 seconds. Select only high resolution plot with auto scaling. The FORTRAN code can be written as:

```
TTYPE=1
TMAGN=1.
TZERO=5.
TEND=25.
TDELT=.5
GRAFP=0
HRDCPY=1
CALL STIME(2)
```

PRECMP DIRECTIVE *STIME i { ttype tmagn tzero tend tdelt }

produces the following FORTRAN statements

```
TTYPE=ttype          "if 1st optional argument is used"
TMAGN=tmagn          "if 2nd optional argument is used"
TZERO=tzero          "if 3rd optional argument is used"
TEND=tend            "if 4th optional argument is used"
TDELT=tdelt          "if 5th optional argument is used"
CALL STIME(i)
```

Example 2: *STIME 2 1 10

produces the following FORTRAN statements

```
TTYPE=0
TMAGN=10
CALL STIME(2)
```

METHOD See Section C.4.

COMMENTS Capability exists for creating multiple plots on the high resolution electrostatic plotter by setting the appropriate value for the parameter CONTP. See Section E.3 for the details.

In previous versions of LCAP2, the parameter TTYPE was called TSTEP.

Restrictions No multiple poles[1] allowed except for those at the origin. Also, the number of zeros must not exceed the number of non-zero poles (See Section C.3).

[1] If a system with multiple poles are modeled as a connection of s plane transfer functions, command BITIME can be used to evaluate the time response since the partial fraction expansion method is not used.

PURPOSE: Compute s to w plane slow-fast multirate transform (ZOH at slower sampling rate)
WPTF$_i$ = slow-fast multirate transform of **SPTF**$_i$

If the input to an s plane transfer function is sampled at a slow rate and the output is sampled at a faster (by an integer multiple) rate, the transfer function of the output is given by:

$$C^{T/n}(z_n) = G^{T/n}(z_n) * E^T(z)$$

where, z = z plane variable at the slow rate
n = integer ratio of input/output sampling period
z_n = z plane variable at the faster rate
$E^T(z)$ = z plane transform of the input at the slower rate
$G(s)$ = s plane transfer function with time delay and optional ZOH
$G^{T/n}(z_n)$ = z plane transform of G(s) at the faster rate
$C^{T/n}(z_n)$ = z plane transform of the output at the faster rate

The equivalent w plane relationship is given by

$$C^{T/n}(w_n) = G^{T/n}(w_n) * E^T(w)$$

where, w = w plane variable at the slow rate
w_n = w plane variable at the faster rate
$E^T(w)$ = w plane transform of the input at the slower rate
$G^{T/n}(w_n)$ = w plane transform of G(s) at the faster rate
$C^{T/n}(w_n)$ = w plane transform of the output at the faster rate

The command SWMRX computes the transform $G^{T/n}(w_n)$.

$G^{T/n}(w_n)$ at the faster sampling rate could be computed using the command SWXFM if there were no slow rate zero order hold at the input to the plant. If there is one, the effects of the delay term of the zero order hold must be correctly accounted for. In using the command SWMRX, the faster sampling rate is specified by the parameter SAMPT and the integer ratio of the input/output sampling period is specified by NTGER. Parameters used by SWMRX are in common block DBASE described by the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
SAMPT	1.	Sampling period of the faster output sampler
NTGER	1	Integer ratio of input/output sampling period
DELAY	0.	Time delay (enter negative value for time advance)
ZOH	1	.NE.0 for inclusion of zero order hold at the slower input rate

FORTRAN CALL CALL SWMRX(i , j)

Example 1: Input to a continuous plant with a zero order hold is sampled at 1 Hz. The plant is modeled as an s plane transfer functior. in $\mathbf{SPTF}_2$ and the output is sampled at 3 Hz. Compute the output transform at 3 Hz assuming that the transform of the input is an impulse. Store the resulting transform into $\mathbf{WPTF}_5$. The FORTRAN code can be written as:

```
TWOPI=6.2831853
SAMPT=TWOPI/3.
NTGER=3
ZOH=1
DELAY=0
CALL SWMRX(5,2)
```

PRECMP DIRECTIVE *SWMRX i j { sampt delay zoh ntger }

produces the following FORTRAN statements

```
SAMPT=sampt     "if the 1st optional argument is used"
DELAY=delay     "if the 2nd optional argument is used"
ZOH=zoh         "if the 3rd optional argument is used"
NTGER=ntger     "if the 4th optional argument is used"
CALL STIME(i)
```

Example 2:

```
*SWMRX 5 2 2.5
```

produces the following FORTRAN statements

```
SAMPT=2.5
CALL SWMRX(5,2)
```

METHODS See Section C.6.

PURPOSE: Compute w transform of an s plane transfer function $\mathbf{SPTF}_i$
$\mathbf{WPTF}_i$ = w transform of $\mathbf{SPTF}_j$

This is the classical w transform computed by the partial fraction method. The transform includes time delay and a zero hold. The parameters used by SWXFM are in common block DBASE described by the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
SAMPT	1.	Sampling period of the faster output sampler
DELAY	0.	Time delay (enter negative value for time advance)
ZOH	1	.NE.0 for inclusion of zero order hold at the slower input rate

FORTRAN CALL CALL SWXFM(i , j)

Example 1: Compute the w transform of a continuous transfer function stored in $\mathbf{SPTF}_3$. The plant includes a zero order hold and a time delay of 0.1 seconds. The sampling period is 0.5 seconds. Store the resulting transform into $\mathbf{WPTF}_1$. The FORTRAN code can be written as:

```
SAMPT=.5
DELAY=.1
ZOH=1
CALL SWXFM(1,3)
```

PRECMP DIRECTIVE *SWXFM i j { sampt delay zoh }

produces the following FORTRAN statements

```
SAMPT=sampt    "if the 1st optional argument is used"
DELAY=delay    "if the 2nd optional argument is used"
ZOH=zoh        "if the 3rd optional argument is used"
CALL SWXFM(i,j)
```

Example 2: `*SWXFM 1 3 .5`

produces the following FORTRAN statements

```
SAMPT=.5
CALL SWXFM(1,3)
```

METHOD The partial fraction method is used to compute the sampled-data transform. See Section C.5.

RESTRICTIONS The algorithms used for the partial fraction expansion require the following constraints on the s plane transfer function: (1) multiple poles are not allowed except for those at the origin, (2) the poles at the origin (including the 1/s from the zero order hold if there is one) must be 5 or less, and (3) the degree of the numerator must not be greater than the number of poles not at the origin.

PURPOSE: Compute the s to z plane slow-fast multirate transform (ZOH at slower sampling rate)
ZPTF$_i$ = slow-fast multirate transform of **SPTF**$_i$

If the input to an s plane transfer function is sampled at a slow rate and the output is sampled at a faster (by an integer multiple) rate, the transfer function of the output is given by:

$$C^{T/n}(z_n) = G^{T/n}(z_n) * E^T(z)$$

where, z = z plane variable at the slow rate
n = integer ratio of input/output sampling period
z_n = z plane variable at the faster rate
$E^T(z)$ = z plane transform of the input at the slower rate
$G(s)$ = s plane transfer function with time delay and optional ZOH
$G^{T/n}(z_n)$ = z plane transform of G(s) at the faster rate
$C^{T/n}(z_n)$ = z plane transform of the output at the faster rate

$G(z_n)$ at the faster sampling rate could be computed using the command SZXFM if there were no slow rate zero order hold at the input to the plant. If there is one, the effects of the delay term of the zero order hold must be correctly accounted for. In using the command SZMRX, the faster sampling rate is specified by the parameter SAMPT and the integer ratio of the input/output sampling period is specified by NTGER. Parameters used by SZMRX are in common block DBASE described by the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
SAMPT	1.	Sampling period of the faster output sampler
NTGER	1	Integer ratio of input/output sampling period
DELAY	0.	Time delay (enter negative value for time advance)
ZOH	1	.NE.0 for inclusion of zero order hold at the slower input rate

FORTRAN CALL CALL SZMRX(i , j)

Example 1: Input to a continuous plant with a zero order hold is sampled at 1 Hz. The plant is modeled as an s plane transfer function in $\mathbf{SPTF_4}$ and is sampled at 3 Hz. Compute the z transform at 3 Hz at the output of the plant assuming that the transform of the input is stored in $\mathbf{ZPTF_1}$. First compute the multirate z transform of the plant with a zero order hold and store it into $\mathbf{ZPTF_5}$. The desired output transform is then $\mathbf{ZPTF_1} * \mathbf{ZPTF_5}$. LCAP2 cannot compute this product since the z variable of $\mathbf{ZPTF_5}$ is at the faster sampling rate and the z variable of $\mathbf{ZPTF_1}$ is at the slower rate. However, the input transform can be expressed at the faster rate by changing its z variable to z_n by using the ZVCNG command. If this were done and $\mathbf{ZPTF_2}$ is the input transform expressed at the faster z_n variable, the desired output transform is the product $\mathbf{ZPTF_2} * \mathbf{ZPTF_5}$. Compute this product and store into $\mathbf{ZPTF_6}$. The FORTRAN code can be written as:

```
TWOPI=6.2831853
SAMPT=TWOPI/3.
NTGER=3
ZOH=1
DELAY=0
CALL SWMRX(5,4)
CALL ZVCNG(2,1)
CALL ZPMPY(6,5,2)
```

PRECMP DIRECTIVE *SZMRX i j { sampt delay zoh ntger }

produces the following FORTRAN statements

```
SAMPT=sampt    "if the 1st optional argument is used"
DELAY=delay    "if the 2nd optional argument is used"
ZOH=zoh        "if the 3rd optional argument is used"
NTGER=ntger    "if the 4th optional argument is used"
CALL SZMRX(i,j)
```

Example 1: `*SZMRX 5 4 .5 .3`

produces the following FORTRAN statements

```
SAMPT=.5
DELAY=.3
CALL SZMRX(5,4)
```

METHOD See Section C.6.

SZXFM

PURPOSE: Compute the z transform of an s plane transfer function $\mathbf{SPTF}_j$
$\mathbf{ZPTF}_i$ = z transform of $\mathbf{SPTF}_j$

This is the classical z transform computed by the partial fraction method. The transform includes time delay and a zero hold. The parameters used by SZXFM are in common block DBASE described by the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
SAMPT	1	Sampling period of the faster output sampler
DELAY	0.	Time delay (enter negative value for time advance)
ZOH	1	.NE.0 for inclusion of zero order hold at the slower input rate

FORTRAN CALL `CALL SZXFM( i , j )`

Example 1: Compute the z transform of a continuous transfer function stored in $\mathbf{SPTF}_3$. The plant includes a zero order hold and a time delay of 0.1 seconds. The sampling period is 0.5 seconds. Store the resulting transform into $\mathbf{ZPTF}_1$. The FORTRAN code can be written as:

```
SAMPT=.5
DELAY=.1
ZOH=1
CALL SZXFM(1,3)
```

PRECMP DIRECTIVE `*SZXFM i j { sampt delay zoh }`

produces the following FORTRAN statements

```
SAMPT=sampt    "if the 1st optional argument is used"
DELAY=delay    "if the 2nd optional argument is used"
ZOH=zoh        "if the 3rd optional argument is used"
CALL SZXFM(i,j)
```

Example 2: `*SZXFM 1 3 .8`

produces the following FORTRAN statements

```
SAMPT=.8
CALL SZXFM(1,3)
```

METHOD The partial fraction method is used to compute the sampled-data transform. For higher accuracy the computations are performed in the w plane and then transformed into the z plane. See Section C.5.

RESTRICTIONS The algorithms used for the partial fraction expansion require the following constraints on the s plane transfer function: (1) multiple poles are not allowed except for those at the origin, (2) the poles at the origin (including the 1/s from the zero order hold if there is one) must be 5 or less, and (3) the degree of the numerator must not be greater than the number of poles not at the origin.

PURPOSE: Eliminate common roots of w plane transfer function **WPTF**$_i$

FORTRAN CALL $\boxed{\text{CALL WELCR(i)}}$

Example 1: Eliminate common roots of **WPTF**$_2$. The FORTRAN code can be written as:

```
CALL WELCR(2)
```

PRECMP DIRECTIVE $\boxed{\text{*WELCR i}}$

produces the following FORTRAN statement

```
CALL WELCR(i)
```

Example 2:

```
*WELCR 2
```

produces the following FORTRAN statement

```
CALL WELCR(2)
```

METHOD If a numerator root nrt and a denominator root drt are found such that (1) CABS(drt/nrt - (1.,0.)) .LT. ECRE1 for nrt .NE. 0. or (2) CABS(drt). LT. ECRE2 for nrt .EQ. 0., roots nrt and drt are considered to be common and will be eliminated from the transfer function. ECRE1 and ECRE2 are in common block DBASE described in the following table. See Section C.8.

COMMON/DBASE/ parameters used		
parameter	preset	description
ECRE1	2.E-4	Tolerance for eliminating common roots in subroutine CRELIM
ECRE2	1.E-8	Tolerance for "zero" root in subroutine CRELIM

PURPOSE: Compute frequency response of w plane transfer function $\mathbf{WPTF}_i$

The frequencies to be used in evaluating the frequency response of $\mathbf{WPTF}_i$ are specified in the s plane for the convenience of the user. The program will make the conversion to the w plane when the response is computed.

Two modes are available for selecting the frequency points used for evaluating the frequency response of transfer function $\mathbf{WPTF}_i$. The automatic mode, selected when FAUTO is nonzero, allows the user to specify up to 20 preselected frequencies with the array OMEGA. In this mode the program will use all these preselected points plus its own dynamically generated points to yield a smooth plot. The number of preselected points is determined by NOMEGA. The beginning and last frequency points for computing the response are determined by OMEGA(1) and OMEGA(NOMEGA), respectively.

In the nonautomatic frequency mode (FAUTO=0) the user can define up to five sets of frequencies to be used in computing the response. Each of these sets is specified by a three element array of the form FREQk(i), i=1,3. If FREQk(1)=a, FREQk(2)=b and FREQk(3)=c, the k-th set of frequencies specified is:

a, a+c, a+2c, ... a+jc, b

where j is the largest integer such that (a+jc) is less than b. Each successive FREQk array must define an increasing set of frequencies such that the first value of the segment is always larger than the last value of the preceding segment. When FREQk(3) is not larger than FREQk(1), as in the case with the preset values for $k = 2,5$, those segments will not be used.

With either the automatic or the nonautomatic frequency mode, the program will automatically check for the gain and phase crossover. When found, the program will iterate until the exact crossover frequency is found. The limit on the number of plot points computed is 1500.

By default the units of OMEGA, FREQ1, FREQ2, FREQ3, FREQ4, and FREQ5 are in rad/sec. However, they can be changed to Hz by setting the parameter RAD to zero.

Bode, Nichols, and Nyquist plots are selected by the values of the flags FBODE, FNICO, FNYQS, respectively. The more commonly used parameters by WFREQ in common block DBASE are given in the following table. They include plot parameters.

If a Bode plot is selected and the user enters a starting frequency 0.0 instead of a nonzero value, the program will plot the response over 6 cycles (ignoring the value of CYCLE) with a starting value equal to 1.E-6*(largest frequency selected).

COMMON/DBASE/ parameters used		
parameter	preset	description
FAUTO	1	.NE.0 for automatic frequency point selection. Uses NOMEGA and array OMEGA. .EQ.0 for user supplied frequency points. Uses arrays FREQ1, FREQ2, ..., FREQ5.
NOMEGA	2	Number of preselected frequency points in array OMEGA for use in auto. frequency mode. (range = 2 - 20)
OMEGA		Array of preselected frequency points for auto. frequency mode. (units determined by RAD)
	1.	OMEGA(1) = first frequency point used in auto. mode
	10.	OMEGA(2) = second frequency point used in auto. mode
		OMEGA(NOMEGA) = last frequency point used in auto. mode
RAD	1	.NE.0 for rad/sec, otherwise Hz
FBODE	1	.NE.0 for Bode plot
FNICO	0	.NE.0 for Nichols plot
PMARG	0	.NE.0 for plotting phase margin instead of phase for the Nichols plot
FNYQS	0	.NE.0 for Nyquist plot
NQDB	0	.NE.0 for hardcopy Nyquist in plot in dB when used with FNYQS
GRAFP	1	.NE.0 for low resolution printer plots
HRDCPY	0	.NE.0 for high resolution electrostatic plots
DEGMN	-360.	Minimum defined phase in frequency response (Phase defined from DEGMN to DEGMN+360.)
CYCLE	0	Number of log cycles for a Bode plot (0-6) .EQ.0 for automatic selection
FREQ1(1)	1.	Starting freq. point for first segment of user specified values. (when FAUTO=0). (Units determined by RAD)
FREQ1(2)	10.	End freq. point for first segment of user specified values (when FAUTO=0).
FREQ1(3)	1.	Delta frequency for first segment of user specified values (when FAUTO=0).
FREQk(1)	0.	Starting freq. point for k-th segment of user specified values (when FAUTO=0). k=2,5
FREQk(2)	0.	End freq. point for k-th segment of user specified values (when FAUTO=0). k=2,5
FREQk(3)	0.	Delta frequency for k-th segment of user specified values (when FAUTO=0). k=2,5
DBMAX	0.	Maximum dB for plotting frequency response
DBMIN	0.	Minimum dB for plotting frequency response (Auto. scaling if DBMIN=DBMAX)
FXYDEL	.5	Nyquist plot scale in units per inch. (Auto scaling if FXYDEL=0)
FXYMIN	-2.5	Nyquist plot parameter - minimum real and imag. value plotted
SAMPT	1.	Sampling period

FORTRAN CALL CALL WFREQ(i)

Example 1: Compute the frequency response of $\mathbf{WPTF_3}$, using the auto frequency mode for frequencies between 0. and 1000. and at 1., 10., and 100. The sampling period of $\mathbf{WPTF_3}$ is 0.2 seconds. Select printer and hardcopy Bode and Nichols plots with auto scaling. The FORTRAN code can be written as:

```
FAUTO=1
NOMEGA=5
OMEGA(1)=0.
OMEGA(2)=1.
OMEGA(3)=10.
OMEGA(4)=100.
OMEGA(5)=1000.
FBODE=1
FNICO=1
GRAFP=1
HRDCPY=1
SAMPT=.2
CALL WFREQ(3)
```

Example 2: Compute the frequency response of $\mathrm{WPTF_3}$, using the nonauto frequency mode for frequencies between 0. and 2. in increments of .1; between 3. and 20. in increments of 1.; and between 22. and 100. in increments of 2. The sampling period of $\mathbf{WPTF_3}$ is 0.2 seconds. Select printer and hardcopy Bode and Nichols plots with fixed scaling between -30 and 10 for the dB axis. The FORTRAN code can be written as:

```
FAUTO=0
FREQ1(1)=0.
FREQ1(2)=2.
FREQ1(3)=.1
FREQ2(1)=3.
FREQ2(2)=20.
FREQ2(3)=1.
FREQ3(1)=22.
FREQ3(2)=100.
FREQ3(3)=2.
FBODE=1
FNICO=1
DBMIN=-30
DBMAX=10
SAMPT=.2
CALL WFREQ(3)
```

PRECMP DIRECTIVE *WFREQ i { sampt }

produces the following FORTRAN statements

```
SAMPT=sampt    "if 1st optional argument is used"
CALL WFREQ(i)
```

Example 3: Using Example 1 from above, the following code fragment

```
      FAUTO=1
      NOMEGA=5
*OMEGA 0 1 10 100 1000
      FBODE=1
      FNICO=1
      GRAFP=1
      HRDCPY=1
      SAMPT=.2
*WFREQ 3
```

produces the same FORTRAN statements as in Example 1.

METHOD The frequency response can be evaluated by using either (1) the coefficient form or (2) the root form (if available) of **WPTF**$_i$. If the roots are available, that form will be used since the response can be computed with higher accuracy.[1]

If the automatic frequency mode is selected (FAUTO.NE.0), the program will choose frequency points for evaluating the response such that successive dB and phase values will be within specified limits to yield a smooth plot. The program evaluates the first point using f = OMEGA(1). Then choosing deltaf = OMEGA(1)/20 initially, the next frequency to be used is computed as f =f + deltaf. Evaluating the response using this value of f, the delta dB and phase are compared to the specified limits. If either is too large, deltaf is halvcd and the response is recomputed. If both are too small, deltaf is doubled and the response is recomputed. The limits for delta dB is EDB1/2 and EDB1. The limits for delta phase is EDEG1/2 and EDEG1. Simultaneously with computing the next f be to used in evaluating the response, a comparison is made with the next value of OMEGA(i). If f is larger than OMEGA(i), f will be replaced with the value of OMEGA(i). This will ensure that the user's prespecified frequency points will be used. This procedure will continue until the last value of OMEGA(NOMEGA) is used. There is a limit on the number of iterations in computing the frequency points. This limit can be changed by the parameter MAXITF.

If a frequency point being used to evaluate the response is equal to a pole on the jω, the warning message

ZERO IN DENOMINATOR AT OMEGA = ___ , THIS POINT SKIPPED

[1] Also, for very high order transfer functions, the following problems will be avoided: (1) Premature overflow of the numerator or denominator terms if the coefficients are scaled by a very large number. (This can occur if a large number of block diagram reduction operations were used to compute **WPTF**$_i$. For this situation, though, the command WNORM can be used to normalize the coefficients.) and (2) Inherently less accurate representation of the coefficients in a finite machine (the coefficients are functions of terms involving products of the roots) will become pronounced as the order of the transfer function is increased.

will be printed out and the program allowed to continue. If this situation occurs when the automatic frequency mode is selected and OMEGA(1)=0. (i.e., when the transfer function has a pole at the origin), the program will be restarted with the first frequency point at OMEGA(2)/1000[1] instead.

Also as part of the automatic frequency mode, a comparison is made on deltaf to keep deltaf/f within the limits of MINDW and MAXDW. The lower limit MINDW is necessary to prevent an excessive number of plot points around frequencies with very low damping coefficients. The upper limit MAXDW will ensure that there are enough points to yield a smooth Bode plot.

Since the plot points computed to generate a smooth plot will, in many cases, be very large, only a portion of the computed response will be printed out. The printout is controlled by the delta dB and delta phase parameters, EDB2 and EDEG2, respectively. A tabular printout is made only if either of these limits is exceeded.

Capability exists for creating multiple plots on the high resolution electrostatic printer by setting the appropriate value for the parameter CONTP. See Section E.3 for the details.

The following table lists additional parameters in the common block DBASE which the user can change.

COMMON/DBASE/ parameters used		
parameter	preset	description
CONTP	0	=0 Single curve plot =1 First curve of a plot =2 Continuation of a plot =3 Final curve of a plot
EDB1	1.	Min. delta dB for plotting
EDB2	2.	Min. delta dB for printout
EDEG1	4.	Max. delta degree for plotting
EDEG2	10.	Max. delta degree of printout
MINDW	.0005	Min. relative frequency step size
MAXDW	.2	Max. relative frequency step size
MAXITF	3000	Max. number of iterations allowed

[1] This will allow a nonzero deltaf = (OMEGA(2)/1000)/20 to be used instead of OMEGA(1)/20 for initializing the variable frequency step size.

PURPOSE: Compute root loci of w plane transfer function $\mathbf{WPTF}_i$

Root locus of an open loop transfer function $\mathbf{WPTF}_i$ is computed by varying the loop gain. The gains to be used by WLOCI are multipliers of $\mathbf{WPTF}_i$. Thus, the gains should begin at a value less than 1.0 and end at a value greater than 1.0 if the loci is to bracket the nominal operating gain of the system. Up to 25 preselected gains for evaluating the root loci can be specified with array KGAIN. The number of preselected gains is determined by NLOCI. The beginning and last gains are determined by KGAIN(1) and KGAIN(NLOCI), respectively. Additional gains can be automatically computed by the program to fill in values between the preselected gains. Between KGAIN(i) and KGAIN(i+1), additional gains will be selected by either of the following two methods if they are between KGAIN(i) and KGAIN(i+1).

If KFLG.EQ.0, gain = KGAIN(i) + KDELT*j, j=1,2, ...

If KFLG.NE.0, gain = KGAIN(i)*KDELT**j, j=1,2, ...

For example, if KGAIN(2)=10, KGAIN(3)=100, KFLG=1, and KDELT=2, the following gains will be used to compute the root loci:

. . . 10., 20., 40., 80., 100., . . .

The total number of gains used is limited by the value of the parameter ITLOC which is preset to 50. Parameters used by WLOCI are given in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
NLOCI	2	Number of root locus gains entered in array KGAIN (max=25)
KGAIN		Array of root locus gains
	.5	KGAIN(1) = first user-specified root locus gain
	2.	KGAIN(2) = second user-specified root locus gain
		KGAIN(NLOCI) = last user-specified root locus gain.
		(Gains computed and used only if they are between KGAIN(1) and KGAIN(NLOCI))
KFLG	1	.EQ.0 to increment gain by multiplying by KDELT .NE.0 to increment gain by adding by KDELT
KDELT	1.E4	Value for changing gains (preset to large value so that no additional gains are computed).
ITLOC	50	Max. number of different gains computed
GRAFP	1	.NE.0 for low resolution printer plots
HRDCPY	0	.NE.0 for high resolution electrostatic plots
RLXMIN	0	Min. x axis for plotting
RLXMAX	0	Max. x axis for plotting (Auto. scaling of x axis if RLXMIN=RLXMAX)
RLYMIN	0	Min. y axis for plotting
RLYMAX	0	Max. y axis for plotting (Auto. scaling of y axis if RLYMIN=RLYMAX)
RLFLG1	1	Flag for numbering root locus points on hardcopy plots .EQ.1 for numbering; .EQ.-1 for no numbering

FORTRAN CALL CALL WLOCI(i)

Example 1: Compute the root locus of **WPTF$_2$** by varying the gain from 0.5 to 20 by doubling the first gain until the last gain is reached. Select printer and hardcopy plots with auto scaling. The FORTRAN code can be written as:

```
NLOCI=2
KGAIN(1)=.5
KGAIN(2)=20.
KFLG=0
KDELT=2
GRAFP=1
HRDCPY=1
CALL WLOCI(2)
```

PRECMP DIRECTIVE *WLOCI i

produces the following FORTRAN statement

```
CALL WLOCI(I)
```

Example 2: Using Example 1 from above, the following code fragment

```
      NLOCI=2
*KGAIN .5 20.
      KDELT=2
      GRAFP=1
      HRDCPY=1
*WLOCI 2
```

produces the following FORTRAN statements

```
      NLOCI=2.
      KGAIN(1)=.5
      KGAIN(2)=20.
      KFLG=0.
      KDELT=2.
      GRAFP=1.
      HRCPY=1.
      CALL WLOCI(2)
```

METHOD Root locus is computed by evaluating the roots of the polynomial (PN + k*PD) where k is the gain to be varied and PN and PD are the numerator and denominator polynomials of the transfer function.

PURPOSE: Compute multirate frequency response of a w plane transfer function $\mathbf{WPTF}_i$ (Frequency decomposition)

If a z transform $G^{T/n}(z_n)$ with period T/n is resampled at a rate which is n times slower, the transform at the slower rate, $G^T(z)$, is given by

$$G^T(z) = \frac{1}{n}\sum_{k=0}^{n-1} G^{T/n}(z_n e^{\frac{j2\pi * k}{n}})$$

The command ZMRFQ computes the frequency response of $G^T(z)$ by evaluating the above equation with shifted values of z_n. The command WMRFQ is the w plane equivalent of ZMRFQ.

The frequency and plot parameters used by WMRFQ are in common block DBASE. Since they are the same ones used by command WFREQ they will not be repeated here (See Reference on WFREQ). The sampling period of the slower rate transform $G^T(w)$ is specified by the parameter SAMPT. The integer ratio of the slow/fast sampling periods is specified by the parameter NTGER. Both of these parameters are in common block DBASE described in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
SAMPT	1.	Sampling period of the slower output transform
NTGER	1	Integer ratio of slow/fast sampling periods

FORTRAN CALL

```
CALL WMRFQ( i )
```

where, i = index of the faster rate w plane transfer function $\mathbf{WPTF}_i$

Example 1: Compute the multirate (slow rate) frequency response of $\mathbf{WPTF}_4$, using the auto frequency mode for frequencies between 0. and 1000. The sampling period of transfer function $\mathbf{WPTF}_4$ is 3 seconds. The output of $\mathbf{WPTF}_4$ is resampled at half the rate of the input, i.e., sampling period of 6 seconds. Select printer and hardcopy Bode plots with auto scaling. The FORTRAN code can be written as:

```
FAUTO=1
NOMEGA=2
OMEGA(1)=0.
OMEGA(2)=1000.
FBODE=1
GRAFP=1
HRDCPY=1
SAMPT=6.
NTGER=2
CALL WMRFQ(4)
```

PRECMP DIRECTIVE *WMRFQ i { sampt ntger }

produces the following FORTRAN statements

```
SAMPT=sampt        "if 1st optional argument is used"
NTGER=ntger        "if 2nd optional argument is used"
CALL WMRFQ(i)
```

Example 2: Using Example 1 from above, the following code fragment

```
      FAUTO=1
      NOMEGA=2
*OMEGA 0 1000
      FBODE=1
      GRAFP=1
      HRDCPY=1
*WMRFQ 4 6 2
```

produces the following FORTRAN statements

```
      FAUTO=1
      NOMEGA=2
      OMEGA(1)=0.
      OMEGA(2)=1000.
      FBODE=1
      GRAFP=1
      HRDCPY=1
      SAMPT=6
      NTGER=2
      CALL WMRFQ(4)
```

COMMENTS There is a command WMRXFM which will compute a rational representation of the slower rate transform G^T(w). However, the numerator roots may not be very accurate if the degree is very large. The command WMRFQ can be used to verify that the transform computed by WMRXFM is correct.

PURPOSE: Multirate (fast-to-slow rate) transform of a w plane transfer function by frequency decomposition

WPTF$_i$ = frequency decomposition of **WPTF**$_j$

If a z transform $G^{T/n}(z_n)$ with period T/n is resampled at a rate which is n times slower, the transform at the slower rate, $G^T(z)$, is given by

$$G^T(z) = \frac{1}{n}\sum_{k=0}^{n-1} G^{T/n}(z_n e^{\frac{j2\pi * k}{n}})$$

The command ZMRXFM computes the rational representation of $G^T(z)$ as a function of the slower rate z variable. The command WMRXFM is the w plane equivalent of ZMRXFM.

The sampling period of the slower rate transform $G^T(w)$ is specified by the parameter SAMPT. The integer ratio of slow/fast sampling periods is specified by the parameter NTGER. Both of these parameters are in common block DBASE described in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
SAMPT	1.	Sampling period of the slower output transform
NTGER	1	Integer ratio of slow/fast sampling periods

FORTRAN CALL CALL WMRXFM(i, j)

where, i = index of the slower rate w plane transform **WPTF**$_i$
j = index of the faster rate w plane transform **WPTF**$_j$

Example 1: Compute the multirate (frequency decomposition) transform of **WPTF**$_5$ and store into **WPTF**$_4$. The sampling period of transform **WPTF**$_5$ is 3 seconds. The output of **WPTF**$_5$ is resampled at one third the rate of the input, i.e., sampling period of 1 second. The FORTRAN code can be written as:

```
SAMPT=1.
NTGER=3
CALL WMRXFM(4,5)
```

PRECMP DIRECTIVE *WMRXFM i j { sampt ntger } produces the following FORTRAN statements

```
SAMPT=sampt     "if 1st optional argument is used"
NTGER=ntger     "if 2nd optional argument is used"
CALL WMRXFM(i,j)
```

Example 2: Using Example 1 from above, the following PRECMP directive

```
*WMRXFM 4 5 1. 3
```

produces the following FORTRAN statements

```
SAMPT=1.
NTGER=3
CALL WMRXFM(4,5,3)
```

METHOD See Section C.6.

COMMENTS The results of this transform may not be very accurate for higher order transforms. To check the accuracy, the frequency response of the resulting transform **WPTF**$_i$ can be evaluated and compared with the frequency response computed by the command WMRFQ.

PURPOSE: Normalize coefficients of w plane transfer function $\mathbf{WPTF}_i$

Normalization can be either with respect to the low order nonzero coefficient or the high order coefficient of the denominator. The parameter NRMHI determines which method is to be used.

If NRMHI is equal to zero, the low order nonzero coefficient of the denominator is set equal to the parameter KNORM and all other coefficients are normalized to this value.

If NRMHI is not equal to zero, the high order coefficient of the denominator is set equal to the parameter KNORM and all other coefficients are normalized to this value. Parameters used by WNORM are given in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
KNORM	1.	Value used for normalizing the transfer function
NRMHI	0	.EQ.0 for normalizing to the low order nonzero coefficient .NE.0 for normalizing to the high order coefficient

FORTRAN CALL CALL WNORM(i)

Example 1: If the w plane transfer function

$$\frac{w + 14}{2w^2 + 7w + 78}$$

stored in $\mathbf{WPTF}_1$, is to be normalized so that the low order denominator coefficient is 1.0, the FORTRAN code can be written as:

```
NRMHI=0
KNORM=1.
CALL WNORM(1)
```

PRECMP DIRECTIVE *WNORM i { nrmhi knorm }

produces the following FORTRAN statements

```
NRMHI=nrmhi    "if 1st optional argument is used"
KNORM=knorm    "if 2nd optional argument is used"
CALL WNORM(i)
```

Example: `*WNORM 1 0 2.`

produces the following FORTRAN statements

```
NRMHI=0
KNORM=2.
CALL WNORM(1)
```

PURPOSE: W plane transfer function add
$\mathbf{WPTF}_i = \mathbf{WPTF}_j + \mathbf{WPTF}_k$

FORTRAN CALL $\boxed{\text{CALL WPADD(i , j , k)}}$

Example: Add $\mathbf{WPTF}_4$ to $\mathbf{WPTF}_2$ and store into $\mathbf{WPTF}_3$. The FORTRAN code can be written as:

```
CALL WPADD(3,4,2)
```

PRECMP DIRECTIVE $\boxed{\text{*WPADD i j k}}$

produces the following FORTRAN statement

```
CALL WPADD(i,j,k)
```

Example: `*WPADD 3 4 2`

produces the following FORTRAN statement

```
CALL WPADD(3,4,2)
```

COMMENTS If the roots of both $\mathbf{WPTF}_j$ and $\mathbf{WPTF}_k$ are available (loaded in or computed from a previous command) any common roots between the denominators will be factored out before the sum is computed.

PURPOSE: W plane closed loop transfer function
$\mathbf{WPTF}_i = \mathbf{WPTF}_j \,/\, (1 + \mathbf{WPTF}_j * \mathbf{WPTF}_k)$

FORTRAN CALL $\boxed{\text{CALL WPCLSLP(i , j , k)}}$

Example: Compute closed loop transfer function $\mathbf{WPTF}_4$ where $\mathbf{WPTF}_2$ is the forward loop transfer function and $\mathbf{WPTF}_3$ is the feedback transfer function. The FORTRAN code can be written as:

```
CALL WPCLSLP(4,2,3)
```

PRECMP DIRECTIVE $\boxed{\text{*WPCLSLP i j k}}$

produces the following FORTRAN statement

```
CALL WPCLSLP(i,j,k)
```

Example: `*WPCLSLP 4 2 3`

produces the following FORTRAN statement

```
CALL WPCLSLP(4,2,3)
```

WPDEL WPDEL

PURPOSE: W plane transfer function delete
Deletes transfer function **WPTF**$_i$ from the current list of transfer function arrays in use

FORTRAN CALL `CALL WPDEL( i )`

Example: Delete **WPTF**$_3$ from the current list of transfer function arrays in use. The FORTRAN code can be written as:

```
CALL WPDEL(3)
```

PRECMP DIRECTIVE `*WPDEL i`

produces the following FORTRAN statement

```
CALL WPDEL(i)
```

COMMENTS When the command SAVE is invoked all polynomials, transfer functions, and matrix data will be saved to a file. At the present time there is no means to select only a subset of this data to be saved. The command WPDEL, however, allows the user to delete w plane transfer functions from the current list of transfer functions in use.

WPDIV WPDIV

PURPOSE: W plane transfer function divide
$\mathbf{WPTF}_i = \mathbf{WPTF}_j \,/\, \mathbf{WPTF}_k$

FORTRAN CALL `CALL WPDIV( i , j , k )`

Example: Divide $\mathbf{WPTF}_4$ by $\mathbf{WPTF}_2$ and store into $\mathbf{WPTF}_3$. The FORTRAN code can be written as:

```
CALL WPDIV(3,4,2)
```

PRECMP DIRECTIVE `*WPDIV  i  j  k`

produces the following FORTRAN statement

```
CALL WPDIV(i,j,k)
```

Example: `*WPDIV 3 4 2`

produces the following FORTRAN statement

```
CALL WPDIV(3,4,2)
```

METHOD If the roots of $\mathbf{WPTF}_j$ and $\mathbf{WPTF}_k$ are available (loaded in or computed in a previous command) the quotient is computed from the roots instead of from the coefficients.

WPEQU

PURPOSE: W plane transfer function equal
$\mathbf{WPTF_i = WPTF_j}$

FORTRAN CALL $\boxed{\text{CALL WPEQU(i , j)}}$

Example: Equate **$\mathbf{WPTF_4}$** to **$\mathbf{WPTF_2}$**. The FORTRAN code can be written as:

```
CALL WPEQU(4,2)
```

PRECMP DIRECTIVE $\boxed{\text{*WPEQU i j}}$

produces the following FORTRAN statements

```
CALL WPEQU(i,j)
```

Example:

```
*WPEQU 4 2
```

produces the following FORTRAN statement

```
CALL WPEQU(4,2)
```

PURPOSE: Load transfer function coefficients into $\mathbf{WPTF}_i$
$\mathbf{WPTF}_i$ = POLYN / POLYD

To load coefficients into $\mathbf{WPTF}_i$, coefficients must first be loaded into the arrays POLYN and POLYD. The command WPLDC copies the coefficients in POLYN and POLYD into $\mathbf{WPTF}_i$. The arrays POLYN and POLYD are in common block POLYCM described in the following table:

COMMON/POLYCM/ parameters used		
parameter	preset	description
POLYN	0	Array for entering coefficients of a numerator polynomial for use with command SPLDC, WPLDC, ZPLDC. The format is: POLYN(1) = degree (.LE. MXPDEG) POLYN(2) = coefficient of **0 POLYN(3) = coefficient of **1 ⋮ POLYN(n+1) = coefficient of **n
POLYD	0	Array for entering coefficients of a denominator polynomial for use with command SPLDC, WPLDC, ZPLDC. The format is: POLYD(1) = degree (.LE. MXPDEG) POLYD(2) = coefficient of **0 POLYD(3) = coefficient of **1 ⋮ POLYD(n+1) = coefficient of **n

FORTRAN CALL `CALL WPLDC( i )`

Example 1: Load the transfer function

$$\frac{8w+3}{17w^2+6w+11}$$

into $\mathbf{WPTF}_4$. The first step is to load the numerator and denominator into POLYN and POLYD, respectively. The FORTRAN code can be written as:

```
POLYN(1)=1
POLYN(2)=3
POLYN(3)=8
POLYD(1)=2
POLYD(2)=11
POLYD(3)=6
POLYD(4)=17
CALL WPLDC(4)
```

PRECMP DIRECTIVE $\boxed{*\text{WPLDC i}}$

produces the following FORTRAN statement

```
CALL WPLDC(i)
```

Example 2: Using Example 1 from above, the following PRECMP directives

```
*POLYN 1 3 8
*POLYD 2 11 6 17
*WPLDC 4
```

produces the same FORTRAN code given in Example 1. See Section 4.2.4 for definition of the *POLYN and *POLYD directives.

RESTRICTIONS The degree of the numerator and denominator must be .LE.MXPDEG. (MXPDEG = 49 for the regular version of LCAP2, = 100 for large model version on the CRAY).

PURPOSE: Load transfer function roots into $\mathbf{WPTF}_i$
$\mathbf{WPTF}_i$ = ROOTN / ROOTD

To load roots into $\mathbf{WPTF}_i$, coefficients must first be loaded into the arrays ROOTN and ROOTD. The command WPLDR copies the coefficients in ROOTN and ROOTD into $\mathbf{WPTF}_i$. The complex arrays ROOTN and ROOTD are in common block ROOTCM described in the following table:

COMMON/ROOTCM/ parameters used		
parameter	preset	description
ROOTN	0	Complex array for entering numerator roots into a transfer function for use with commands SPLDC, WPLDC, or ZPLDC. The format is: ROOTN(1) = CMPLX(degree,gainn) where gainn = low order nonzero numerator coefficient ROOTN(2) = numerator root number 1 ROOTN(3) = numerator root number 2 ⋮ ROOTN(n+1) = numerator root number n
ROOTD	0	Complex array for entering numerator roots into a transfer function for use with commands SPLDC, WPLDC, or ZPLDC. The format is: ROOTD(1) = CMPLX(degree,gaind) where gaind = low order nonzero denominator coefficient ROOTD(2) = denominator root number 1 ROOTD(3) = denominator root number 2 ⋮ ROOTD(n+1) = denominator root number n

FORTRAN CALL CALL WPLDR(i)

Example 1: Load the transfer function

$$\frac{50(\frac{w}{2}+1)}{7(\frac{w}{4}+1)(\frac{w}{3-j5}+1)(\frac{w}{3+j5}+1)}$$

into $\mathbf{WPTF}_2$. The first step is to load the numerator and denominator into the complex arrays ROOTN and ROOTD, respectively. The FORTRAN code can be written as:

```
ROOTN(1)=(1.,50.)
ROOTN(2)=(-2.,0.)
ROOTD(1)=(3.,7.)
ROOTD(2)=(-4.,0.)
ROOTD(3)=(-3.,5.)
ROOTD(4)=(-3.,-5.)
CALL WPLDR(2)
```

PRECMP DIRECTIVE *WPLDR i j

produces the following FORTRAN statement

```
CALL WPLDR(i)
```

Example 2: Using Example 1 from above, the following PRECMP directives

```
*ROOTN 50. -2
*ROOTD 7 -4 (-3.,5.)
*WPLDR 2
```

will yield a set of FORTRAN statements which will be functionally equivalent to those given in Example 1 above. The FORTRAN code generated by the *ROOTN and *ROOTD PRECMP directives includes calls to several subroutines which will convert various root formats (i.e. real, (real,imag), [zeta,omega], and <tau>) into LCAP2 root array format. Note that the number of roots do not have to be entered. It will be computed by the *ROOTN and *ROOTD directives. Also, it can optionally allow the user to specify the high order coefficients instead of the low order nonzero coefficients for the numerator and denominator "gains". See Section 4.2.4 for description of *ROOTN and *ROOTD.

RESTRICTIONS The degree of the numerator and denominator must be .LE. MXPDEG. (MXPDEG = 49 for the regular version of LCAP2, = 100 for the large order model on the CRAY).

WPMPY WPMPY

PURPOSE: W plane transfer function multiply
$\mathbf{WPTF}_i = \mathbf{WPTF}_j * \mathbf{WPTF}_k$

FORTRAN CALL CALL WPMPY(i , j , k)

Example: Multiply $\mathbf{WPTF}_4$ and $\mathbf{WPTF}_2$ and store into $\mathbf{WPTF}_3$. The FORTRAN code can be written as:

```
CALL WPMPY(3,4,2)
```

PRECMP DIRECTIVE *WPMPY i j k

produces the following FORTRAN statement

```
CALL WPMPY(i,j,k)
```

Example: `*WPMPY 3 4 2`

produces the following FORTRAN statement

```
CALL WPMPY(3,4,2)
```

METHOD If the roots of $\mathbf{WPTF}_j$ and $\mathbf{WPTF}_k$ are available (loaded in or computed in a previous command) the product is computed from the roots instead of from the coefficients.

WPTF
alias: **WPPRN**

WPTF
alias: **WPPRN**

PURPOSE: Print out s plane transfer function $\mathbf{WPTF}_i$

FORTRAN CALL CALL WPTF(i)

Example: Print out contents of $\mathbf{WPTF}_1$. The FORTRAN code can be written as:

```
CALL WPTF(1)
```

PRECMP DIRECTIVE *WPTF i

produces the following FORTRAN statement

```
CALL WPTF(i)
```

Example: `*WPTF 1`

produces the following FORTRAN statement

```
CALL WPTF(1)
```

COMMENTS This command was called WPPRN in previous versions of LCAP2.

WPRTS WPRTS

PURPOSE: Compute roots of w plane transfer function $\mathbf{WPTF}_i$

FORTRAN CALL CALL WPRTS(i)

Example 1: Compute the roots of $\mathbf{WPTF}_2$. The FORTRAN code can be written as:

```
CALL WPRTS(2)
```

PRECMP DIRECTIVE *WPRTS i

produces the following FORTRAN statement

```
CALL WPRTS(i)
```

Example 2:

```
*WPRTS 2
```

produces the following FORTRAN statement

```
CALL WPRTS(2)
```

METHOD The Aerospace root finding MULE [8] routine is used to find the roots. See Section C.1.

PURPOSE: W plane transfer function subtract
$\mathbf{WPTF}_i = \mathbf{WPTF}_j - \mathbf{WPTF}_k$

FORTRAN CALL CALL WPSUB(i , j , k)

Example: Subtract $\mathbf{WPTF}_4$ from $\mathbf{WPTF}_2$ and store into $\mathbf{WPTF}_3$. The FORTRAN code can be written as:

```
CALL WPSUB(3,2,4)
```

PRECMP DIRECTIVE *WPSUB i j k

produces the following FORTRAN statement

```
CALL WPSUB(i,j,k)
```

Example: `*WPSUB 3 2 4`

produces the following FORTRAN statement

```
CALL WPSUB(3,2,4)
```

COMMENTS If the roots of both $\mathbf{WPTF}_j$ and $\mathbf{WPTF}_k$ are available (loaded in or computed from a previous command) any common roots between the denominators will be factored out before the difference is computed.

WSXFM WSXFM

PURPOSE: Transform w plane roots to "equivalent" s plane roots

Transform w plane roots of $\mathbf{WPTF}_i$ into "equivalent" s plane roots. The transformation of w plane roots to the s plane is not unique. The "equivalent" s plane roots are provided solely to aid the analyst in identifying and correlating w plane roots. The computed roots are not saved into an s-plane transfer function. The sampling period of $\mathbf{WPTF}_i$ is specified by the parameter SAMPT in common block DBASE described by the following table.

COMMON/DBASE/		parameters used
parameter	preset	description
SAMPT	1.	Sampling period

FORTRAN CALL `CALL WSXFM( i )`

where, i = index of w plane transfer function $\mathbf{WPTF}_i$

Example 1: Compute the "equivalent" s plane roots of $\mathbf{WPTF}_3$ which has a sampling period of 1.5 seconds. The FORTRAN code can be written as:

```
SAMPT=1.5
CALL WSXFM(3)
```

PRECMP DIRECTIVE `*WSXFM i { sampt }`

produces the following FORTRAN statements

```
SAMPT=sampt      "if 1st optional argument is used"
CALL WSXFM(i)
```

Example 2:

```
*WSXFM 3 1.5
```

produces the following FORTRAN statements

```
SAMPT=1.5
CALL WSXFM(3)
```

METHOD Transformation of the roots from w to the s plane is defined by

$$s = \ln(\ (1+w)/(1-w)\ )\ /\ SAMPT$$

When w = -1.0 or +1.0 the "equivalent' s plane root is undefined. If CABS(1.−w) is less than 1.E-5, the equivalent root is printed out as 999999.99. If CABS(w+1.) is less than 1.E-5, the equivalent root is printed out as -999999.99.

PURPOSE: Bilinear transform from w to z plane
$\mathbf{ZPTF}_i$ = z transform of $\mathbf{WPTF}_j$

Compute the bilinear transform of $\mathbf{WPTF}_j$ and store into $\mathbf{ZPTF}_i$. The bilinear transform is defined by,

$$z = \frac{1+w}{1-w}$$

FORTRAN CALL `CALL WZXFM( i )`

where, i = index of z plane transfer function $\mathbf{ZPTF}_i$
j = index of w plane transfer function $\mathbf{WPTF}_j$

Example 1: Compute the w to z plane bilinear transform of $\mathbf{WPTF}_2$ and store into $\mathbf{ZPTF}_5$. The FORTRAN code can be written as:

```
CALL WZXFM(5,2)
```

PRECMP DIRECTIVE `*WZXFM i j`

produces the following FORTRAN statement

```
CALL WZXFM(i,j)
```

Example 2:

```
*WZXFM 5 2
```

produces the following FORTRAN statement

```
CALL WZXFM(5,2)
```

METHOD For higher accuracy, the bilinear transform is implemented by using the roots of $\mathbf{WPTF}_i$ rather than its coefficients.

PURPOSE: Eliminate common roots of z plane transfer function $\mathbf{ZPTF}_i$

FORTRAN CALL CALL ZELCR(i)

Example 1: Eliminate common roots of $\mathbf{ZPTF}_2$. The FORTRAN code can be written as:

```
CALL ZELCR(2)
```

PRECMP DIRECTIVE *ZELCR i

produces the following FORTRAN statement

```
CALL ZELCR(i)
```

Example 2:

```
*ZELCR 2
```

produces the following FORTRAN statement

```
CALL ZELCR(2)
```

METHOD If a numerator root nrt and a denominator root drt are found such that (1) CABS(drt/nrt - (1.,0.)) .LT. ECRE1 for nrt .NE. 0. or (2) CABS(drt). LT. ECRE2 for nrt .EQ. 0., roots nrt and drt are considered to be common and will be eliminated from the transfer function. ECRE1 and ECRE2 are in common block DBASE described in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
ECRE1	2.E-4	Tolerance for eliminating common roots in subroutine CRELIM
ECRE2	1.E-8	Tolerance for "zero" root in subroutine CRELIM

See Section C.8.

PURPOSE: Compute frequency response of z plane transfer function $\mathbf{ZPTF}_i$

The frequencies to be used in evaluating the frequency response of $\mathbf{ZPTF}_i$ are specified in the s plane for the convenience of the user. The program will make the conversion to the z plane when the response is computed.

Two modes are available for selecting the frequency points used for evaluating the frequency response of transfer function $\mathbf{ZPTF}_i$. The automatic mode, selected when FAUTO is non-zero, allows the user to specify up to 20 preselected frequencies with the array OMEGA. In this mode the program will use all these preselected points plus its own dynamically generated points to yield a smooth plot. The number of preselected points is determined by NOMEGA. The beginning and last frequency points for computing the response are determined by OMEGA(1) and OMEGA(NOMEGA), respectively.

In the nonautomatic frequency mode (FAUTO=0) the user can define up to five sets of frequencies to be used in computing the response. Each of these sets are specified by a three element array of the form FREQk(i), i=1,3. If FREQk(1)=a, FREQk(2)=b and FREQk(3)=c, the k-th set of frequencies specified is:

a, a+c, a+2c, ... a+jc, b

where j is the largest integer such that (a+jc) is less than b. Each successive FREQk array must define an increasing set of frequencies such that the first value of the segment is always larger than the last value of the preceding segment. When FREQk(3) is not larger than FREQk(1), as in the case with the preset values for $k = 2, 5$, those segments will not be used.

With either the automatic or the nonautomatic frequency mode, the program will automatically check for the gain and phase crossover. When found, the program will iterate until the exact crossover frequency is found. The limit on the number of plot points computed is 1500.

By default the units of OMEGA, FREQ1, FREQ2, FREQ3, FREQ4, and FREQ5 are in rad/sec. However, they can be changed to Hz by setting the parameter RAD to zero.

Bode, Nichols, and Nyquist plots are selected by the values of the flags FBODE, FNICO, FNYQS, respectively. The more commonly used parameters by ZFREQ in common block DBASE are given in the following table. They include plot parameters.

If a Bode plot is selected and the user enters a starting frequency of 0.0 instead of a nonzero value, the program will plot the response over 6 cycles (ignoring the value of CYCLE) with a starting value equal to 1.E-6*(largest frequency selected).

COMMON/DBASE/ parameters used

parameter	preset	description
FAUTO	1	.NE.0 for automatic frequency point selection. Uses NOMEGA and array OMEGA. .EQ.0 for user supplied frequency points. Uses arrays FREQ1, FREQ2, ..., FREQ5.
NOMEGA	2	Number of preselected frequency points in array OMEGA for use in auto. frequency mode. (range = 2 - 20)
OMEGA		Array of preselected frequency points for auto. frequency mode. (units determined by RAD)
	1.	OMEGA(1) = first frequency point used in auto. mode
	10.	OMEGA(2) = second frequency point used in auto. mode
		OMEGA(NOMEGA) = last frequency point used in auto. mode
RAD	1	.NE.0 for rad/sec, otherwise Hz
FBODE	1	.NE.0 for Bode plot
FNICO	0	.NE.0 for Nichols plot
PMARG	0	.NE.0 for plotting phase margin instead of phase for the Nichols plot
FNYQS	0	.NE.0 for Nyquist plot
NQDB	0	.NE.0 for hardcopy Nyquist plot in dB when used with FNYQS
GRAFP	1	.NE.0 for low resolution printer plots
HRDCPY	0	.NE.0 for high resolution electrostatic plots
DEGMN	-360.	Minimum defined phase in frequency response (Phase defined from DEGMN to DEGMN+360.)
CYCLE	0	Number of log cycles for a Bode plot (0-6) .EQ.0 for automatic selection
FREQ1(1)	1.	Starting freq. point for first segment of user specified values (when FAUTO=0). (Units determined by RAD)
FREQ1(2)	10.	End freq. point for first segment of user specified values (when FAUTO=0).
FREQ1(3)	1.	Delta frequency for first segment of user specified values (when FAUTO=0).
FREQk(1)	0.	Starting freq. point for k-th segment of user specified values (when FAUTO=0). k=2,5
FREQk(2)	0.	End freq. point for k-th segment of user specified values (when FAUTO=0). k=2,5
FREQk(3)	0.	Delta frequency for k-th segment of user specified values (when FAUTO=0). k=2,5
DBMAX	0.	Maximum dB for plotting frequency response
DBMIN	0.	Minimum dB for plotting frequency response. (Auto. scaling if DBMIN=DBMAX)
FXYDEL	.5	Nyquist plot scale in units per inch. (Auto scaling if FXYDEL=0)
FXYMIN	-2.5	Nyquist plot parameter - minimum real and imag. value plotted
SAMPT	1.	Sampling period

FORTRAN CALL CALL ZFREQ(i)

Example 1: Compute the frequency response of **ZPTF**$_3$, using the auto frequency mode for frequencies between 0. and 1000. and at 1., 10., and 100. The sampling period of **ZPTF**$_3$ is 0.2 seconds. Select printer and hardcopy Bode and Nichols plots with auto scaling. The FORTRAN code can be written as:

```
FAUTO=1
NOMEGA=5
OMEGA(1)=0.
OMEGA(2)=1.
OMEGA(3)=10.
OMEGA(4)=100.
OMECA(5)=1000.
FBODE=1
FNICO=1
GRAFP=1
HRDCPY=1
SAMPT=.2
CALL ZFREQ(3)
```

Example 2: Compute the frequency response of **ZPTF**$_3$, using the nonauto frequency mode for frequencies between 0. and 2. in increments of 0.1; between 3. and 20. in increments of 1.; and between 22. and 100. in increments of 2. The sampling period of **ZPTF**$_3$ is 0.2 seconds. Select printer and hardcopy Bode and Nichols plots with fixed scaling between -30 and 10 for the dB axis. The FORTRAN code can be written as:

```
FAUTO=0
FREQ1(1)=0.
FREQ1(2)=2.
FREQ1(3)=.1
FREQ2(1)=3.
FREQ2(2)=20.
FREQ2(3)=1.
FREQ3(1)=22.
FREQ3(2)=100.
FREQ3(3)=2.
FBODE=1
FNICO=1
DBMIN=-30
DBMAX=10
SAMPT=.2
CALL ZFREQ(3)
```

PRECMP DIRECTIVE $\boxed{\text{*ZFREQ i \{ sampt \}}}$

produces the following FORTRAN statements:

```
SAMPT=sampt         "if 1st optional argument is used"
CALL ZFREQ(i)
```

Example 3: Using Example 1 from above, the following code fragment

```
      FAUTO=1
      NOMEGA=5
*OMEGA 0 1 10 100 1000
      FBODE=1
      FNICO=1
      GRAFP=1
      HRDCPY=1
      SAMPT=.2
*ZFREQ 3
```

produces the same FORTRAN statements as in Example 1.

METHOD The frequency response can be evaluated by using either (1) the coefficient form or (2) the root form (if available) of $\mathbf{ZPTF}_i$. If the roots are available, that form will be used since the response can be computed with higher accuracy.[1]

If the automatic frequency mode is selected (FAUTO.NE.0), the program will choose frequency points for evaluating the response such that successive dB and phase values will be within specified limits to yield a smooth plot. The program evaluates the first point using f = OMEGA(1). Then choosing deltaf = OMEGA(1)/20 initially, the next frequency to be used is computed as f =f + deltaf. Evaluating the response using this value of f, the delta dB and phase is compared to the specified limits. If either is too large, deltaf is halved and the response is recomputed. If both are too small, deltaf is doubled and the response is recomputed. The limits for delta dB is EDB1/2 and EDB1. The limits for delta phase is EDEG1/2 and EDEG1. Simultaneously with computing the next f be to used in evaluating the response, a comparison is made with the next value of OMEGA(i). If f is larger than OMEGA(i), f will be replaced with the value of OMEGA(i). This will ensure that the user's prespecified frequency points will be used. This procedure will continue until the last value of OMEGA(NOMEGA) is used. There is a limit on the number of iterations in computing the frequency points. This limit can be changed by the parameter MAXITF.

If a frequency point being used to evaluate the response is equal to a pole on the unit circle, the warning message

[1] Also, for very high order transfer functions, the following problems will be avoided: (1) Premature overflow of the numerator or denominator terms if the coefficients are scaled by a very large number. (This can occur if a large number of block diagram reduction operations were used to compute $\mathbf{ZPTF}_i$. For this situation, though, the command ZNORM can be used to normalize the coefficients.) and (2) Inherently less accurate representation of the coefficients in a finite machine (the coefficients are functions of terms involving products of the roots) will become pronounced as the order of the transfer function is increased. For z plane transfer functions, the coefficient form representation is generally useless when the order is greater than 15~20.

ZERO IN DENOMINATOR AT OMEGA = ..., THIS POINT SKIPPED

will be printed out and the program allowed to continue. If this situation occurs when the automatic frequency mode is selected and OMEGA(1)=0. (i.e., when the transfer function has a pole at z = 1.) the program will be restarted with the first frequency point at OMEGA(2)/1000[1] instead.

Also as part of the automatic frequency mode, a comparison is made on deltaf to keep deltaf/f within the limits of MINDW and MAXDW. The lower limit MINDW is necessary to prevent an excessive number of plot points around frequencies with very low damping coefficients. The upper limit MAXDW will ensure that there are enough points to yield a smooth Bode plot.

Since the plot points computed to generate a smooth plot will, in many cases, be very large, only a portion of the computed response will be printed out. The printout is controlled by the delta dB and delta phase parameters, EDB2 and EDEG2, respectively. A tabular printout is made only if either of these limits are exceeded.

Capability exists for creating multiple plots on the high resolution electrostatic printer by setting the appropriate value for the parameter CONTP. See Section E.3 for the details.

The following table lists additional parameters in the common block DBASE which the user can change.

COMMON/DBASE/ parameters used		
parameter	preset	description
CONTP	0	=0 Single curve plot =1 First curve of a plot =2 Continuation of a plot =3 Final curve of a plot
EDB1	1.	Min. delta dB for plotting
EDB?	2.	Min. delta dB for printout
EDEG1	4.	Max. delta degree for plotting
EDEG2	10.	Max. delta degree of printout
MINDW	.0005	Min. relative frequency step size
MAXDW	.2	Max. relative frequency step size
MAXITF	3000	Max. number of iterations allowed

[1] This will allow a nonzero deltaf = (OMEGA(2)/1000)/20 to be used instead of OMEGA(1)/20 for initializing the variable frequency step size.

PURPOSE: Compute root loci of z plane transfer function **ZPTF**$_i$

Root locus of an open loop transfer function **ZPTF**$_i$ is computed by varying the loop gain. The gains to be used by ZLOCI are multipliers of **ZPTF**$_i$. Thus, the gains should begin at a value less than 1.0 and end at a value greater than 1.0 if the loci is to bracket the nominal operating gain of the system. Up to 25 preselected gains for evaluating the root loci can be specified with array KGAIN. The number of preselected gains is determined by NLOCI. The beginning and last gains are determined by KGAIN(1) and KGAIN(NLOCI), respectively. Additional gains can be automatically computed by the program to fill in values between the preselected gains. Between KGAIN(i) and KGAIN(i+1), additional gains will be selected by either of the following two methods if they are between KGAIN(i) and KGAIN(i+1).

If KFLG.EQ.0, gain = KGAIN(i) + KDELT*j, j=1,2, ...

If KFLG.NE.0, gain = KGAIN(i)*KDELT**j, j=1,2, ...

For example, if KGAIN(2)=10, KGAIN(3)=100, KFLG=1, and KDELT=2, the following gains will be used to compute the root loci:

. . . 10., 20., 40., 80., 100., . . .

The total number of gains used is limited by the value of the parameter ITLOC which is preset to 50. Parameters used by ZLOCI are given in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
NLOCI	2	Number of root locus gains entered in array KGAIN (max=25)
KGAIN		Array of root locus gains
	.5	KGAIN(1) = first user-specified root locus gain
	2.	KGAIN(2) = second user-specified root locus gain
		KGAIN(NLOCI) = last user-specified root locus gain.
		(Gains computed and used only if they are between KGAIN(1) and KGAIN(NLOCI))
KFLG	1	.EQ.0 to increment gain by multiplying by KDELT .NE.0 to increment gain by adding by KDELT
KDELT	1.E4	Value for changing gains (preset to large value so that no additional gains are computed).
ITLOC	50	Max. number of different gains computed
GRAFP	1	.NE.0 for low resolution printer plots
HRDCPY	0	.NE.0 for high resolution electrostatic plots
RLXMIN	0.	Min. x axis for plotting
RLXMAX	0.	Max. x axis for plotting (Auto. scaling of x axis if RLXMIN=RLXMAX)
RLYMIN	0.	Min. y axis for plotting
RLYMAX	0.	Max. y axis for plotting (Auto. scaling of y axis if RLYMIN=RLYMAX)
RLFLG1	1	Flag for numbering root locus points on hardcopy plots .EQ.1 for numbering; .EQ.-1 for no numbering

FORTRAN CALL CALL ZLOCI(i)

Example 1: Compute the root locus of $\mathbf{WPTF_2}$ by varying the gain from 2 to 20 by doubling the first gain until the last gain is reached. Select printer and hardcopy plots with auto scaling. The FORTRAN code can be written as:

```
NLOCI=2
KGAIN(1)=2
KGAIN(2)=20
KFLG=0
KDELT=2
GRAFP=1
HRDCPY=1
CALL WLOCI(2)
```

PRECMP DIRECTIVE *ZLOCI i

produces the following FORTRAN statements

```
CALL ZLOCI(I)
```

Example 2: Using Example 1 from above, the following code fragment

```
      NLOCI=2
*KGAIN 2 20
      KFLG=0
      KDELT=2
      GRAFP=1
      HRDCPY=1
*ZLOCI 2
```

produces the following FORTRAN statements

```
      NLOCI=2.
      KGAIN(1)=2.
      KGAIN(2)=20.
      KFLG=0.
      KDELT=2.
      GRAFP=1.
      HRDCPY=1.
      CALL ZLOCI(2)
```

METHOD Root locus is computed by evaluating the roots of the polynomial (PN + k*PD) where k is the varied gain and PN and PD are the numerator and denominator polynomials of the transfer function.

PURPOSE: Compute multirate frequency response of a z plane transfer function **ZPTF**$_i$ (Frequency decomposition)

If a z transform $G^{T/n}(z_n)$ with period T/n is resampled at a rate which is n times slower, the transform at the slower rate, $G^T(z)$, is given by

$$G^T(z) = \frac{1}{n}\sum_{k=0}^{n-1} G^{T/n}(z_n e^{\frac{j2\pi * k}{n}})$$

The command ZMRFQ computes the frequency response of $G^T(z)$ by evaluating the above equation with shifted values of z_n.

The frequency and plot parameters used by ZMRFQ are in common block DBASE. Since they are the same ones used by command ZFREQ they will not be repeated here (See Reference on ZFREQ). The sampling period of the slower rate transform $G^T(z)$ is specified by the parameter SAMPT. The integer ratio of the slow/fast sampling periods is specified by the parameter NTGER. Both of these parameters are in common block DBASE described in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
SAMPT	1.	Sampling period of the slower output transform
NTGER	1	Integer ratio of slow/fast sampling periods

FORTRAN CALL CALL ZMRFQ(i)

where, i = index of the faster rate z plane transfer function **ZPTF**$_i$

Example 1: Compute the multirate (slow rate) frequency response of **ZPTF**$_4$, using the auto frequency mode for frequencies between 0. and 1000. The sampling period of transfer function **ZPTF**$_4$ is 3 seconds. The output of **ZPTF**$_4$ is resampled at half the rate of the input, i.e., sampling period of 6 seconds. Select printer and hardcopy Bode plots with auto scaling. The FORTRAN code can be written as:

```
FAUTO=1
NOMEGA=2
OMEGA(1)=0.
OMEGA(2)=1000.
FBODE=1
GRAFP=1
HRDCPY=1
SAMPT=6.
NTGER=2
CALL ZMRFQ(4)
```

PRECMP DIRECTIVE *ZMRFQ i { sampt ntger }

produces the following FORTRAN statements

```
SAMPT=sampt      "if 1st optional argument is used"
NTGER=ntger      "if 2nd optional argument is used"
CALL ZMRFQ(i,n)
```

Example 2: Using Example 1 from above, the following code fragment

```
      FAUTO=1
      NOMEGA=2
*OMEGA 0 1000
      FBODE=1
      GRAFP=1
      HRDCPY=1
*ZMRFQ 4 6 2
```

produces the following FORTRAN statements

```
      FAUTO=1
      NOMEGA=2
      OMEGA(1)=0.
      OMEGA(2)=1000.
      FBODE=1
      GRAFP=1
      HRDCPY=1
      SAMPT=6
      NTGER=2
      CALL ZMRFQ(4)
```

COMMENTS There is a command ZMRXFM which will compute a rational representation of the slower rate transform $G^T(z)$. However, the numerator roots may not be very accurate if the degree is very large. The command ZMRFQ can be used to verify that the transform computed by ZMRXFM is correct.

PURPOSE: Multirate (fast-to-slow rate) transform of a z plane transfer function by frequency decomposition
ZPTF$_i$ = frequency decomposition of **ZPTF**$_j$

If a z transform $G^{T/n}(z_n)$ with period T/n is resampled at a rate which is n times slower, the transform at the slower rate, $G^T(z)$, is given by

$$G^T(z) = \frac{1}{n}\sum_{k=0}^{n-1} G^{T/n}(z_n e^{\frac{j2\pi*k}{n}})$$

The command ZMRXFM computes the rational representation of $G^T(z)$ as a function of the slower rate z variable.

The sampling period of the slower rate transform $G^T(z)$ is specified by the parameter SAMPT. The integer ratio of slow/fast sampling periods is specified by the parameter NTGER. Both of these parameters are in common block DBASE described in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
SAMPT	1.	Sampling period of the slower output transform
NTGER	1	Integer ratio of slow/fast sampling periods

FORTRAN CALL

```
CALL ZMRXFM( i, j )
```

where, i = index of the slower rate z plane transform **ZPTF**$_i$
j = index of the faster rate z plane transform **ZPTF**$_j$

Example 1: Compute the multirate (frequency decomposition) transform of **ZPTF**$_5$ and store into **ZPTF**$_4$. The sampling period of transform **ZPTF**$_5$ is 3 seconds. The output of **ZPTF**$_5$ is resampled at one third the rate of the input, i.e., sampling period of 1 seconds. The FORTRAN code can be written as:

```
SAMPT=1.
NTGER=3
CALL ZMRXFM(4,5)
```

PRECMP DIRECTIVE

```
*ZMRXFM i j { sampt ntger }
```

produces the following FORTRAN statements

```
SAMPT=sampt    "if 1st optional argument is used"
NTGER=ntger    "if 2nd optional argument is used"
CALL ZMRXFM(i,j)
```

Example 2: Using Example 1 from above, the following PRECMP directive

```
*ZMRXFM 4 5 1. 3
```

produces the following FORTRAN statements

```
SAMPT=1.
NTGER=3
CALL ZMRXFM(4,5,3)
```

METHOD See Section C.6.

COMMENT The results of this transform may not be very accurate for higher order transforms. To check the accuracy, the frequency of the resulting transform $\mathbf{ZPTF}_i$ can be evaluated and compared with the frequency response computed by the command ZMRFQ.

PURPOSE: Normalize coefficients of z plane transfer function **ZPTF**$_i$

Normalization can be either with respect to the low order nonzero coefficient or the high order coefficient of the denominator. The parameter NRMHI determines which method is to be used

If NRMHI is equal to zero, the low order nonzero coefficient of the denominator is set equal to the parameter KNORM and all other coefficients are normalized to this value.

If NRMHI is not equal to zero, the high order coefficient of the denominator is set equal to the parameter KNORM and all other coefficients are normalized to this value. Parameters used by ZNORM are given in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
KNORM	1.	Value used for normalizing the transfer function
NRMHI	0	.EQ.0 for normalizing to the low order non-zero coefficient .NE.0 for normalizing to the high order coefficient

FORTRAN CALL `CALL ZNORM( i )`

Example 1: If the z plane transfer function

$$\frac{3.4z + 3.2}{2z + 1.3}$$

stored in **ZPTF**$_1$, is to be normalized so that the high order denominator coefficient is 1.0, the FORTRAN code can be written as:

```
NRMFG=1
KNORM=1.
CALL ZNORM(1)
```

PRECMP DIRECTIVE `*ZNORM i { nrmhi knorm }`

produces the following FORTRAN statements

```
NRMHI=nrmhi    "if 1st optional argument is used"
KNORM=knorm    "if 2nd optional argument is used"
CALL ZNORM(i)
```

Example:

```
*ZNORM 1 0 2.
```

produces the following FORTRAN statements

```
NRMHI=0
KNORM=2.
CALL ZNORM(1)
```

PURPOSE: Z plane transfer function add
$ZPTF_i = ZPTF_j + ZPTF_k$

FORTRAN CALL `CALL ZPADD( i , j , k )`

Example: Add **$ZPTF_4$** to **$ZPTF_2$** and store into **$ZPTF_3$**. The FORTRAN code can be written as:

```
CALL ZPADD(3,4,2)
```

PRECMP DIRECTIVE `*ZPADD i j k`

produces the following FORTRAN statement

```
CALL ZPADD(i,j,k)
```

Example: `*ZPADD 3 4 2`

produces the following FORTRAN statement

```
CALL ZPADD(3,4,2)
```

COMMENTS If the roots of both **$ZPTF_j$** and **$ZPTF_k$** are available (loaded in or computed from a previous command) any common roots between the denominators will be factored out before the sum is computed.

ZPCLSLP

PURPOSE: Z plane closed loop transfer function

$\mathbf{ZPTF}_i = \mathbf{ZPTF}_j \;/\; (1 + \mathbf{ZPTF}_j * \mathbf{ZPTF}_k)$

FORTRAN CALL CALL ZPCLSLP(i , j , k)

Example: Compute closed loop transfer function $\mathbf{ZPTF}_4$ where $\mathbf{ZPTF}_2$ is the forward loop transfer function and $\mathbf{ZPTF}_3$ is the feedback transfer function. The FORTRAN code can be written as:

```
CALL ZPCLSLP(4,2,3)
```

PRECMP DIRECTIVE *ZPCLSLP i j k

produces the following FORTRAN statement

```
CALL ZPCLSLP(i,j,k)
```

Example: `*ZPCLSLP 4  2 3`

produces the following FORTRAN statement

```
CALL ZPCLSLP(4,2,3)
```

PURPOSE: Z plane transfer function delete
Deletes transfer function $\mathbf{ZPTF}_i$ from the current list of transfer function arrays in use

FORTRAN CALL CALL ZPDEL(i)

Example: Delete $\mathbf{ZPTF}_3$ from the current list of transfer function arrays in use. The FORTRAN code can be written as:

```
CALL ZPDEL(3)
```

PRECMP DIRECTIVE *ZPDEL i

produces the following FORTRAN statement

```
CALL ZPDEL(i)
```

COMMENTS When the command SAVE is invoked all polynomials, transfer functions, and matrix data will be saved to a file. At the present time there is no means to select only a subset of this data to be saved. The command ZPDEL, however, allows the user to delete z plane transfer functions from the current list of transfer functions in use.

PURPOSE: Z plane transfer function divide
$\mathbf{ZPTF}_i = \mathbf{ZPTF}_j / \mathbf{ZPTF}_k$

FORTRAN CALL $\boxed{\text{CALL ZPDIV(i , j , k)}}$

Example: Divide $\mathbf{ZPTF}_4$ by $\mathbf{ZPTF}_2$ and store into $\mathbf{ZPTF}_3$. The FORTRAN code can be written as:

```
CALL ZPDIV(3,4,2)
```

PRECMP DIRECTIVE $\boxed{\text{*ZPDIV i j k}}$

produces the following FORTRAN statement

```
CALL ZPDIV(i,j,k)
```

Example: `*ZPDIV 3 4 2`

produces the following FORTRAN statement

```
CALL ZPDIV(3,4,2)
```

METHOD If the roots of $\mathbf{ZPTF}_j$ and $\mathbf{ZPTF}_k$ are available (loaded in or computed in a previous command) the quotient is computed from the roots instead of from the coefficients.

PURPOSE: Z plane transfer function equal
$ZPTF_i = ZPTF_j$

FORTRAN CALL $\boxed{\text{CALL ZPEQU(i , j)}}$

Example: Equate **$ZPTF_4$** to **$ZPTF_2$**. The FORTRAN code can be written as:

```
CALL ZPEQU(4,2)
```

PRECMP DIRECTIVE $\boxed{\text{*ZPEQU i j}}$

produces the following FORTRAN statement

```
CALL ZPEQU(i,j)
```

Example: `*ZPEQU 4 2`

produces the following FORTRAN statement

```
CALL ZPEQU(4,2)
```

ZPLDC ZPLDC

PURPOSE: Load transfer function coefficients into $\mathbf{ZPTF}_i$
$\mathbf{ZPTF}_i$ = POLYN / POLYD

To load coefficients into $\mathbf{ZPTF}_i$, coefficients must first be loaded into the arrays POLYN and POLYD. The command WPLDC copies the coefficients in POLYN and POLYD into $\mathbf{ZPTF}_i$. The arrays POLYN and POLYD are in common block POLYCM described in the following table:

COMMON/POLYCM/ parameters used		
parameter	preset	description
POLYN	0	Array for entering coefficients of a numerator polynomial for use with command SPLDC, WPLDC, ZPLDC. The format is: POLYN(1) = degree (.LE. MXPDEG) POLYN(2) = coefficient of **0 POLYN(3) = coefficient of **1 ⋮ POLYN(n+1) = coefficient of **n
POLYD	0	Array for entering coefficients of a denominator polynomial for use with command SPLDC, WPLDC, ZPLDC. The format is: POLYD(1) = degree (.LE. MXPDEG) POLYD(2) = coefficient of **0 POLYD(3) = coefficient of **1 ⋮ POLYD(n+1) = coefficient of **n

FORTRAN CALL `CALL ZPLDC( i )`

Example 1: Load the transfer function

$$\frac{1.6z + 1.2}{83z^2 + 160z + 77}$$

into $\mathbf{ZPTF}_4$. The first step is to load the numerator and denominator into POLYN and POLYD, respectively. The FORTRAN code can be written as:

```
POLYN(1)=1
POLYN(2)=1.2
POLYN(3)=1.6
POLYD(1)=2
POLYD(2)=77.
POLYD(3)=160.
POLYD(4)=83.
CALL ZPLDC(4)
```

PRECMP DIRECTIVE *ZPLDC i

produces the following FORTRAN statement

```
CALL ZPLDC(i)
```

Example 2: Using Example 1 from above, the following PRECMP directives

```
*POLYN 1 1.2 1.6
*PCLYD 2 77. 160. 83.
*ZPLDC 4
```

produce the same FORTRAN code given in Example 1. See Section 4.2.4 for definition of the *POLYN and *POLYD directives.

RESTRICTIONS The degree of the numerator and denominator must be .LE.MXPDEG. (MXPDEG = 49 for the regular version of LCAP2, = 100 for large model version on the CRAY).

PURPOSE: Load transfer function roots into $\mathbf{ZPTF}_i$
$\mathbf{ZPTF}_i$ = ROOTN / ROOTD

To load roots into $\mathbf{ZPTF}_i$, coefficients must first be loaded into the arrays ROOTN and ROOTD. The command WPLDR copies the coefficients in ROOTN and ROOTD into $\mathbf{ZPTF}_i$. The complex arrays ROOTN and ROOTD are in common block ROOTCM described in the following table:

COMMON/ROOTCM/ parameters used		
parameter	preset	description
ROOTN	0	Complex array for entering numerator roots into a transfer function for use with commands SPLDC, WPLDC, or ZPLDC. The format is: ROOTN(1) = CMPLX(degree,gainn) where gainn = low order nonzero numerator coefficient ROOTN(2) = numerator root number 1 ROOTN(3) = numerator root number 2 ⋮ ROOTN(n+1) = numerator root number n
ROOTD	0	Complex array for entering numerator roots into a transfer function for use with commands SPLDC, WPLDC, or ZPLDC. The format is: ROOTD(1) = CMPLX(degree,gaind) where gaind = low order nonzero denominator coefficient ROOTD(2) = denominator root number 1 ROOTD(3) = denominator root number 2 ⋮ ROOTD(n+1) = denominator root number n

FORTRAN CALL

```
CALL ZPLDR( i )
```

Example 1: Load the transfer function

$$\frac{2.1(\frac{z}{-.5}+1)}{70(\frac{z}{-.9}+1)(\frac{z}{-.6-j.1}+1)(\frac{z}{-.6+j.1}+1)}$$

into $\mathbf{ZPTF}_2$. The first step is to load the numerator and denominator into the complex arrays ROOTN and ROOTD, respectively. The FORTRAN code can be written as:

```
ROOTN(1)=(1.,2.1)
ROOTN(2)=(.5,0.)
ROOTD(1)=(3.,70.)
ROOTD(2)=(.9,0.)
ROOTD(3)=(.6,.1)
ROOTD(4)=(.6,-.1)
CALL ZPLDR(2)
```

PRECMP DIRECTIVE *ZPLDR i j

produce the following FORTRAN statement

```
CALL ZPLDR(i)
```

Example 2: Using Example 1 from above, the following PRECMP directives

```
*ROOTN 2.1 .5
*ROOTD 70 .9 (.6,.1)
*ZPLDR 2
```

will yield a set of FORTRAN statements which will be functionally equivalent to those given in Example 1 above. The FORTRAN code generated by the *ROOTN and *ROOTD PRECMP directives includes calls to several subroutines which will convert various root formats (i.e. real, (real,imag), [zeta,omega], and <tau>) into LCAP2 root array format. Note that the number of roots do not have to be entered. It will be computed by the *ROOTN and *ROOTD directives. Also, it can optionally allow the user to specify the high order coefficients instead of the low order nonzero coefficients for the numerator and denominator "gains". See Section 4.2.4 for description of *ROOTN and *ROOTD.

RESTRICTIONS The degree of the numerator and denominator must be .LE. MXPDEG. (MXPDEG = 49 for the regular version of LCAP2, = 100 for the large order model on the CRAY).

ZPMPY

PURPOSE: Z plane transfer function multiply
$ZPTF_i$ = $ZPTF_j$ * $ZPTF_k$

FORTRAN CALL CALL ZPMPY(i , j , k)

Example: Multiply **$ZPTF_4$** and **$ZPTF_2$** and store into **$ZPTF_3$**. The FORTRAN code can be written as:

```
CALL ZPMPY(3,4,2)
```

PRECMP DIRECTIVE *ZPMPY i j k

produces the following FORTRAN statement

```
CALL ZPMPY(i,j,k)
```

Example: `*ZPMPY 3 4 2`

produces the following FORTRAN statement

```
CALL ZPMPY(3,4,2)
```

METHOD If the roots of **$ZPTF_j$** and **$ZPTF_k$** are available (loaded in or computed in a previous command) the product is computed from the roots instead of from the coefficients.

ZPTF
alias: **ZPPRN**

PURPOSE: Print out z plane transfer function $\mathbf{ZPTF}_i$

FORTRAN CALL $\boxed{\text{CALL ZPTF(i)}}$

Example: Print out contents of $\mathbf{ZPTF}_1$. The FORTRAN code can be written as:

```
CALL ZPTF(1)
```

PRECMP DIRECTIVE $\boxed{\text{*ZPTF i}}$

produces the following FORTRAN statement

```
CALL ZPTF(i)
```

Example: `*ZPTF 1`

produces the following FORTRAN statement

```
CALL ZPTF(1)
```

COMMENTS This command was called ZPPRN in previous versions of LCAP2.

PURPOSE: Compute roots of z plane transfer function $\mathbf{ZPTF}_i$

FORTRAN CALL CALL ZPRTS(i)

Example 1: Compute the roots of $\mathbf{ZPTF}_2$. The FORTRAN code can be written as:

```
CALL ZPRTS(2)
```

PRECMP DIRECTIVE *ZPRTS i

produces the following FORTRAN statement

```
CALL ZPRTS(i)
```

Example 2:

```
*ZPRTS 2
```

produces the following FORTRAN statement

```
CALL ZPRTS(2)
```

METHOD The Aerospace root finding MULE [8] subroutine is used to find the roots. See Section C.1.

PURPOSE: Z plane transfer function subtract
$\mathbf{ZPTF}_i = \mathbf{ZPTF}_j - \mathbf{ZPTF}_k$

FORTRAN CALL CALL ZPSUB(i , j , k)

Example: Subtract $\mathbf{ZPTF}_4$ from $\mathbf{ZPTF}_2$ and store into $\mathbf{ZPTF}_3$. The FORTRAN code can be written as:

```
CALL ZPSUB(3,2,4)
```

PRECMP DIRECTIVE *ZPSUB i j k

produces the following FORTRAN statement

```
CALL ZPSUB(i,j,k)
```

Example: `*ZPSUB 3 2 4`

produces the following FORTRAN statement

```
CALL ZPSUB(3,2,4)
```

COMMENTS If the roots of both $\mathbf{ZPTF}_j$ and $\mathbf{ZPTF}_k$ are available (loaded in or computed from a previous command) any common roots between the denominators will be factored out before the difference is computed.

PURPOSE: Transform z plane roots to "equivalent" s plane roots

Transform z plane roots of $\mathbf{ZPTF}_i$ into "equivalent" s plane roots. The transformation of z plane roots to the s plane is not unique. The "equivalent" s plane roots are provided solely to aid the analyst in identifying and correlating z plane roots. The computed roots are not saved into a s-plane transfer function. The sampling period of $\mathbf{ZPTF}_i$ is specified by the parameter SAMPT in common block DBASE described by the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
SAMPT	1.	Sampling period

FORTRAN CALL CALL ZSXFM(i)

where, i = index of w plane transfer function $\mathbf{ZPTF}_i$

Example 1: Compute the "equivalent" s plane roots of $\mathbf{ZPTF}_3$ which has a sampling period of 1.5 seconds. The FORTRAN code can be written as:

```
SAMPT=1.5
CALL ZSXFM(3)
```

PRECMP DIRECTIVE *ZSXFM i { sampt }

produces the following FORTRAN statement

```
SAMPT=sampt        "if 1st optional argument is used"
CALL ZSXFM(i)
```

Example 2:

```
*ZSXFM 3
```

produces the following FORTRAN statement

```
CALL ZSXFM(3)
```

METHOD Transformation of the roots from z to the s plane is defined by

$$s = \ln(z) / \text{SAMPT}$$

When z=0 the "equivalent" s plane root is undefined. If CABS(z) is less than 1.E-5, the equivalent s plane root is printed out as -999999.99.

PURPOSE: Time response of transfer function $\mathbf{ZPTF}_i$

Compute time response of a z plane transfer function. The input can be a step or impulse as specified by the parameter TTYPE. The magnitude is specified by the parameter TMAGN. The response will be evaluated every SAMPT seconds from t=0 to t=TEND. Parameters used by ZTIME are in common block DBASE described in the following table.

COMMON/DBASE/ parameters used		
parameter	preset	description
SAMPT	1.	Sampling period
TTYPE	1	.EQ.0 for impulse response; .EQ.1 for step response
TMAGN	1.	Magnitude of input
TEND	1.	End time for evaluating time response
GRAFP	1	.NE.0 for low resolution printer plots
HRDCPY	0	.NE.0 for high resolution electrostatic plots
TXMIN	0.	Minimum x axis for time plot
TXMAX	0.	Maximum x axis for time plot (Auto scaling of x axis if TXMIN=TXMAX)
TYMIN	0.	Minimum y axis for time plot
TYMAX	0.	Maximum y axis for time plot (Auto scaling of y axis if TYMIN=TYMAX)
CONTP	0	=0 Single curve plot =1 First curve of a plot =2 Continuation of a plot =3 Final curve of a plot

FORTRAN CALL CALL ZTIME(i)

Example 1: Compute the step response of $\mathbf{ZPTF}_2$ from 0 to 25 seconds. The sampling period is 0.5 seconds. Select only high resolution hardcopy plot with auto scaling. The FORTRAN code can be written as:

```
SAMPT=.5
TTYPE=1
TMAGN=1.
TEND=25.
GRAFP=0
HRDCPY=1
CALL ZTIME(2)
```

PRECMP DIRECTIVE *ZTIME i { sampt ttype tmagn tend }

produces the following FORTRAN statements

```
SAMPT=sampt       "if 1st optional argument is used"
TTYPE=ttype       "if 2nd optional argument is used"
TMAGN=tmagn       "if 3rd optional argument is used"
TEND=tend         "if 4th optional argument is used"
CALL ZTIME(i)
```

Example 2: `*ZTIME 2 .5 1 1. 25.`

produces the following FORTRAN statement

```
SAMPT=.5
TTYPE=1
TMAGN=1.
TEND=25.
CALL ZTIME(2)
```

METHOD The time response is computed by recursive evaluation of a difference equation. This method in not as accurate as the partial fraction method. However, the results for typical transfer functions whose order is less than 10~15 are very good. The partial fraction method will be implemented at a later date.

COMMENTS Capability exists for creating multiple plots on the high resolution electrostatic printer by setting the appropriate value for the parameter CONTP. See Section E.3 for the details.

In previous versions of LCAP2, the parameter TTYPE was called TSTEP.

ZVCNG

PURPOSE: Z to z^n variable change of a z plane transfer function, i.e.,
$G^{T/n}(z_n) = G^T(z)$, where $z=e^{sT}$ and $z_n=e^{sT/n}$
$\mathbf{ZPTF}_i = G^{T/n}(z_n)$, if $\mathbf{ZPTF}_j = G^T(z)$

The z plane variable of $\mathbf{ZPTF}_j$ is changed from z to z^n for use in performing block diagram reduction of multirate systems. For example,

$$G^{T/n}(z_n) * H^T(z)$$

is the product of two transfer functions at different sampling rates. This product can be simplified if the z variable in $H^T(z)$ were changed to z_n so that the product can be computed by the command ZPMPY. For example, if $H^T(z) = (z+.5)/(z-.9)$, then for n=2, $H^{T/2}(z_2)$ would be $(z_2^2+.5)/(z_2^2-.9)$.

The ratio n is specified by the parameter NTGER in common block DBASE described by the following table

COMMON/DBASE/ parameters used		
parameter	preset	description
NTGER	1	Integer ratio of slow/fast sampling periods

FORTRAN CALL `CALL ZVCNG( i , j )`

Example 1: A plane transfer function with a sampling period of 6 seconds is in $\mathbf{ZPTF}_1$ and a transfer function with a sampling period of 2 seconds is in $\mathbf{ZPTF}_5$. Compute the product of these two transfer functions at the faster sampling rate. The first step will be to use the command ZVCNG to compute an equivalent transfer function to $\mathbf{ZPTF}_1$ but at the faster sampling rate. This transfer function will be stored in $\mathbf{ZPTF}_2$. The last step will be to compute the product of $\mathbf{ZPTF}_2$ and $\mathbf{ZPTF}_5$ and then store it into $\mathbf{ZPTF}_4$. The FORTRAN code can be written as:

```
NTGER=3
CALL ZVCNG(2,1)
CALL ZMPY(4,2,5)
```

PRECMP DIRECTIVE `*ZVCNG i j { ntger }`

produces the following FORTRAN statements

```
NTGER=ntger      "if 1st optional argument is used"
CALL ZVCNG(i,j)
```

Example 2: `*ZVCNG 4 3 2`

produces the following FORTRAN statements

```
NTGER=2
CALL ZTIME(4,3)
```

METHOD The variable change is made by simply copying the coefficients of **ZPTF**$_j$ into the appropriate coefficients of **ZPTF**$_i$. Since the coefficient form of representing a z plane transfer function is only accurate when the order is low, care must taken when using this command. A more accurate implementation of this command would involve computing the nth root of each root in **ZPTF**$_j$.

COMMENTS Since the order of **ZPTF**$_i$ will be n times the order of **ZPTF**$_j$, the user should keep in mind that the maximum size of an LCAP2 transfer function can easily be exceeded.

PURPOSE: Bilinear transform from z to w plane
WPTF$_i$ = w transform of **ZPTF**$_j$

Compute the bilinear transform of **ZPTF**$_j$ and store into **WPTF**$_i$. The bilinear transform is defined by,

$$w = \frac{z-1}{z+1}$$

FORTRAN CALL $\boxed{\text{CALL ZWXFM(i)}}$

where, i = index of z plane transfer function **WPTF**$_i$
j = index of w plane transfer function **ZPTF**$_j$

Example 1: Compute the z to w plane bilinear transform of **WPTF**$_2$ and store into **ZPTF**$_5$. The FORTRAN code can be written as:

```
CALL ZWXFM(5,2)
```

PRECMP DIRECTIVE $\boxed{\text{*ZWXFM i j}}$

produces the following FORTRAN staiement

```
CALL ZWXFM(i,j)
```

Example 2:

```
*ZWXFM 5 2
```

produces the following FORTRAN statement

```
CALL ZWXFM(5,2)
```

METHOD For higher accuracy, the bilinear transform is implemented by using the roots of **ZPTF**$_i$ rather than its coefficients.

Appendix B

Description of LCAP2 Parameters

Param.	Preset	Description
B0	0.	Input vector for coefficients of s^0 term of $\mathbf{B}(s)$ used by command DTERM
B1	0.	Input vector for coefficients of s^1 term of $\mathbf{B}(s)$ used by command DTERM
B2	0.	Input vector for coefficients of s^2 term of $\mathbf{B}(s)$ used by command DTERM
B3	0.	Input vector for coefficients of s^3 term of $\mathbf{B}(s)$ used by command DTERM
B4	0.	Input vector for coefficients of s^4 term of $\mathbf{B}(s)$ used by command DTERM
CONTP	0	Hardcopy plot continuation flag (cannot be used for Bode plots) .EQ.0 - Single plot .EQ.1 - First curve of a plot .EQ.2 - Continuation of a plot .EQ.3 - Final curve of a plot
CYCLE	0	Number of log cycles for a Bode plot (0-6) .EQ.0 for automatic selection
DBMAX	0.	Maximum dB for plotting frequency response
DBMIN	0.	Minimum dB for plotting frequency response (Auto. scaling if DBMIN=DBMAX)
DEGMN	-360.	Minimum defined phase in frequency response (Phase defined from DEGMN to DEGMN+360.)
DELAY	0.	Delay time for sampled-data transform - (sec.)
ECRE1	2.E-4	Tolerance for common root in common root elimination in subroutine CRELIM
ECRE2	1.E-8	Tolerance for zero root in subroutine CRELIM
EDB1	1.	Min. delta dB in frequency response for plotting
EDB2	2.	Min. delta dB in frequency response for printout
EDEG1	4.	Max. delta degree in frequency response for plotting
EDEG2	10.	Max. delta degree in frequency response for printout
EPAD1	1.E-10	Tolerance for negligible higher order coefficients in commands PADD, PSUB, SPADD, SPSUB, WPADD, WPSUB, ZPADD, ZPSUB
EP1	1.E-8	An input quantity in subroutine MULE
EP2	1.E-6	An input quantity in subroutine MULE
EP3	.01	An input quantity in subroutine MULE
EP4	1.E-8	An input quantity in subroutine MULE

Param.	Preset	Description
EP5	1.E-5	An input quantity in subroutine MULE
ERCNJ	1.E-4	Tolerance for zero root in subroutine CRELIM
ERCX	1.E4	Roots are considered to be complex if |imag/real| .GT. ERCX
ERCZ	1.E-5	Roots are considered to be zero if CABS.LT.ERCZ
FAUTO	1	.NE.0 for automatic frequency point selection. Uses NOMEGA and array OMEGA. .EQ.0 for user supplied frequency points. Uses arrays FREQ1, FREQ2, ..., FREQ5.
FBODE	1	.NE.0 for Bode plot with frequency response commands
FDELAY	0.	Time delay (dead time) for use with s plane frequency response
FNICO	0	.NE.0 for Nichols plot with frequency response commands
FNYQS	0	.NE.0 for Nyquist plot with frequency response commands
FREQ1(1)	1.	Starting freq. point for first segment of user specified values (when FAUTO=0). (units determined by RAD)
FREQ1(2)	10.	End freq. point for first segment of user specified values (when FAUTO=0).
FREQ1(3)	1.	Delta frequency for first segment of user specified values (when FAUTO=0).
FREQk(1)	0.	Starting freq. point for k-th segment of user specified values (when FAUTO=0). k=2,5
FREQk(2)	0.	End freq. point for k-th segment of user specified values (when FAUTO=0). k=2,5
FREQk(3)	0.	Delta frequency for k-th segment of user specified values (when FAUTO=0). k=2,5
FXYDEL	.5	Nyquist plot scale in units per inch. (Auto scaling if FXYDL=0)
FXYMIN	-2.5	Nyquist plot parameter - minimum real and imag. value plotted
GRAFP	1	.NE.0 for low resolution printer plots
HRDCPY	0	.NE.0 for high resolution electrostatic plots
ITLOC	50	Max number of different gains computed in root locus
KDELT	1.E4	Value for changing root locus gains (preset to large value so that no additional gains are computed).
KFLG	1	Flag for computing root locus gains .EQ.0 to increment gain by multiplying by KDELT .NE.0 to increment gain by adding by KDELT
KGAIN		Array of root locus gains
	.5	KGAIN(1) = first user-specified root locus gain
	2.	KGAIN(2) = second user-specified root locus gain
		. .
		KGAIN(NLOCI) = last user-specified root locus gain. (Gains computed and used only if they are between KGAIN(1) and KGAIN(NLOCI))
KNORM	1.	Value used for normalizing a transfer function with commands SNORM, WNORM, or ZNORM
MAXDW	.2	Max. relative frequency step size in frequency response when FAUTO.NE.0

Param.	Preset	Description
MAXIT	80	Max number of iterations allowed per root in subroutine MULE
MAXITF	3000	Max. number of iterations in auto. mode of frequency response
MDEG	0	Degree of highest order polynomial in matrices M0, ..., M4 (0-4)
MINDW	.0005	Min. relative frequency step size in frequency response when FAUTO.NE.0
MXM	1	Order of matrices M0, M1, M2, M3, and M4
M0	0.	Matrix for coefficients of s^0 term of M(s) used by commands DETRM and DTERM
M1	0.	Matrix for coefficients of s^1 term of M(s) used by commands DETRM and DTERM
M2	0.	Matrix for coefficients of s^2 term of M(s) used by commands DTERM and DTERM
M3	0.	Matrix for coefficients of s^3 term of M(s) used by commands DETRM and DTERM
M4	0.	Matrix for coefficients of s^4 term of M(s) used by commands DETRM and DTERM
NANOT	4	Number of lines of annotations on hardcopy plots (0-4)
NLOCI	2	Number of root locus gains entered in array KGAIN (2 - 25)
NOMEGA	2	Number of preselected frequency points in array OMEGA for use in auto. frequency mode. (2 - 20)
NP	2	Flag for determining output from subroutine MULE .EQ.0 - print all iterants and BCI output for special procedures .EQ.1 - print only the final iteration of each root .EQ.2 - suppress all internal printing (note - program always print final iteration if max iteration reached
NQDB	0	.NE.0 for hardcopy Nyquist plot in dB when used with FNYQS
NRMHI	0	Transfer function normalization flag, used with SNORM, WNORM, ZNORM .EQ.0 for normalizing to the low order nonzero coefficient .NE.0 for normalizing to the high order coefficient
NTGER	1	Integer n for multirate configuration
OMEGA		Array of pre-selected frequency points for auto. frequency mode. (units determined by RAD)
	1.	OMEGA(1) = first frequency point used in auto. mode
	10.	OMEGA(2) = second frequency point used in auto. mode . . OMEGA(NOMEGA) = last frequency point used in auto. mode
PMARG	0	.NE.0 for plotting phase margin instead of phase for the Nichols plot
POLYP	0.	Array used to input coefficients for polynomials
POLYD	0.	Array used to input denominator coefficients for transfer functions
POLYN	0.	Array used to input numerator coefficients for transfer functions
PRNFLG1	1	(not used)
PRNFLG2	1	.EQ.0 to suppress printout of arguments of an LCAP2 command
PRNFLG3	1	.EQ.0 to suppress printout of an LCAP2 command
PRNFLG4	1	.EQ.0 to suppress printout of results of an LCAP2 command

Param.	Preset	Description
PRNFLG5	1	(not used)
PRNMTRX	0	.EQ.0 for suppressing printout of state space matrices computed by commands B1EIG, B1FREQ, B1LOCI, B1TF, B2EIG, B2FREQ, B2LOCI, and B2TF .EQ.1 for printout of state space matrices .EQ.2 above plus intermediate matrices
PRN1	0	.EQ.0 for suppressing printout from CRELIM
PRN2	0	.EQ.0 for suppressing printout from —
PRN3	0	.EQ.0 for suppressing printout from RCLAS
PRN4	0	.EQ.0 for suppressing printout from RESDU
PRN5	0	.EQ.0 for suppressing printout from —
PRN6	0	.EQ.0 for suppressing printout from WTRANS
PRN7	0	.EQ.0 for suppressing printout from PROOT and MROOT1
PRN8	0	.EQ.0 for suppressing printout from —
PRN9	0	.EQ.0 for suppressing printout from QRRTS and QZRTS
RAD	1	.NE.0 for rad/sec, otherwise Hz
RLFLG1	1	Flag for numbering root locus points on hardcopy plots .EQ.1 for numbering; .EQ.-1 for no numbering
RLXMAX	0.	Max. x axis for plotting with root locus commands
RLXMIN	0.	Min. x axis for plotting with root locus commands (Auto. scaling if RLXMIN=RLXMAX)
RLYMAX	0.	Max. y axis for plotting with root locus commands
RLYMIN	0.	Min. y axis for plotting with root locus commands (Auto. scaling if RLYMIN=RLYMAX)
ROOTP	0.	Complex array used to input roots into polynomials
ROOTD	0.	Complex array used to input denominator roots into transfer functions
ROOTN	0.	Complex array used to input numerator roots into transfer functions
RTMAX	1.E7	Max. root to be found by subroutine MULE
RZERO	1.E-5	Roots .LT. RZERO returned from subroutine MULE are set to 0.
SAMPT	1.	Sampling period (sec.)
SHADE	16	Plot intensity of hardcopy plot (2 - 28)
TDELT	1.	Delta time for s-plane time response
TEND	1.	End time for time response calculation
TMAGN	1.	Magnitude of input for time response calculation
TTYPE	1	.EQ.0 for impulse response; .EQ.1 for step response
TXMAX	0.	Maximum x axis for plotting with time response commands
TXMIN	0.	Minimum x axis for plotting with time response commands (auto scaling of x axis if TXMIN=TXMAX)
TYMAX	0.	Maximum y axis for plotting with time response commands
TYMIN	0.	Minimum y axis for plotting with time response commands (auto scaling of y axis if TYMIN=TYMAX)
TZERO	0.	Start time for evaluating s plane time response
UCIN	0	$\mathbf{C}_i$ block number where input u is connected to (see commands B1FREQ and B1TF)
UDIN	0	$\mathbf{D}_i$ block number where input u is connected to (see commands B2FREQ and B2TF)

Param.	Preset	Description
UMAGN	1.	Magnitude of the input into block $\mathbf{C}_i$ or $\mathbf{D}_i$ (see commands B1FREQ, B1TF, B2FREQ, and B2TF)
XGAP	500	Plot "pen" lifts up when delta x distance exceeds XGAP*(.01) inches
XLINES	52	Number of printer lines used for low resolution printer plots
XNCOL	103	Number of characters used in horizontal direction for printer plots (73 or 103)
XLINES	52	Number of printer lines used for low resolution printer plots
XNCOL	103	Number of characters used in horizontal direction for printer plots (73 or 103)
YANOT	9.6	Y position for second line of hardcopy plot annotation - (in.) (range is 0-10)
YCOUT	0	$\mathbf{C}_i$ block number defining the output for commands B1FREQ and B1TF
YDOUT	0	$\mathbf{D}_i$ block number defining the output for commands B2FREQ and B2TF
YGAP	500	Plot "pen" lifts up when delta y distance exceeds YGAP*(.01) inche
ZLINE	19	Plot intensity for accenting zero axis, normal = 16
ZOH	1	.NE.0 to include zero-order hold in computation of sampled-data transform .EQ.0 to exclude zero-order hold

Appendix C

Mathematical Techniques

C.1 Root Finding

The Aerospace MULE [9] general root finding subroutine is used to solve roots of polynomials and determinants of polynomial matrices. In the abstract of Ref. 9, MULE is described as: "An improved version of Muller's method has been combined with Aitken's delta square extrapolation, an automatic scaling procedure, and a simple but efficient searching technique to provide a general purpose FORTRAN subroutine for the calculation of zeros of arbitrary functions (i.e., F(z) = 0.) using arbitrary guesses. Most of the fundamental problems related to finding zeros of arbitrary functions such as determining initial guesses, convergence of roots, relative zero, absolute zero, 0/0, multiple roots, multiple roots at the origin as well as preventing overflows and underflows have been resolved in a satisfactory manner. ...."

The description of MULE provided below is intended to outline techniques and features which are used by LCAP2. Specific numerical techniques such as extrapolation, scaling, and searching will not be covered.

The roots of an arbitrary function F(z) are found by iteration. This function can be written in product form as,

$$F(z) = k \sum_{i=1}^{n} (z - z_i) \tag{C.1}$$

where z_i are the roots to be found, n is the number of roots, and k is a constant.

To find the first root, MULE uses F(z) directly to determine the iterants. To find the remaining roots, MULE uses the deflated function,

$$F_r(z) = \frac{F(z)}{\sum_{i=1}^{r-1}(z - z_i)} \tag{C.2}$$

to determine the iterants. The index r is the number of the current root to be determined. Since $F_r(z)$ is the original function with previously computed roots factored out, functions with multiple roots or close roots can be more accurately determined by this technique.

The MULE subroutine provides for numerous options and error criteria. Options have been selected to provide the following:

- The user does not have to provide initial guesses for the roots.
- Automatic scaling internal to MULE is provided to handle most problems where the roots are in the range of 1.E-35 < $| z_i |$ < 1.E35.
- The maximum number of iterations is determined by the parameter MAXIT which is preset to 80. When all the roots of a function cannot be found, increasing this parameter may help in some cases.
- No printout of each iteration is provided; however, the user can request this option by changing the value of NP which is described in Appendix B.
- After all roots have been found, the routine may take a second pass to recompute the roots again to improve their accuracy. This occurs if *any* of the roots required the maximum number of iterations (each root max iterated will always be printed out). If a second pass is required, the ordering of the roots from the first pass is reversed and used as guesses for the second pass. If all roots are then successfully found on the second pass, the comment "IGNORE ABOVE WARNING MESSAGE. ALL ROOTS WERE FOUND ON THE 2ND PASS" will be printed out, otherwise, the comment "...OF THE ABOVE ROOTS WERE NOT FOUND ON THE 2ND PASS TO DESIRED PRECISION" will be printed instead.
- Complex roots are found in pairs since the routine automatically uses the conjugate of a root as the next guess if a complex root is found.

One of the arguments in the call to subroutine MULE is the maximum number of roots to be found. When the number of roots found by MULE reaches this value, the root searching procedure is terminated. For functions in which the number of roots are known this value is used for the maximum number of roots to be found. For functions in which the number of roots are not known, an upper limit is used for the value of the maximum number of roots to be found and the technique described below is used to find the exact number of roots.

The searching technique used to find a root starts at or near the origin and proceeds away from the origin until it finds a root. Since $F_r(z)$ instead of F(z) is used to determine the iterants, $F_r(z)$ will become a constant after all the roots have been found. This fact is used to terminate the root finding procedure. After the n^{th} root of F(z) is found MULE will try to find another root. However, since $F_r(z)$ is now a constant, the searching technique will select iterants of successively larger values. Although the exact method of terminating the root finding procedure is more complex, suffice to state that when the iterants exceeds a fixed value, all roots are considered to be found. This fixed value is RTMAX which is preset to 1.0E7. The user can change this value.

As stated above MULE is used to solve roots of polynomials and determinants of polynomial matrices. Since MULE is a general subroutine requiring a separate auxiliary subroutine to evaluate F(z), subroutines PROOT and MROOT1 are used to call subroutine MULE to find, respectively, roots of polynomials and determinants of polynomial matrices.

C.1.1 PROOT

In PROOT the maximum number of roots to be found by MULE is set equal to the degree of the polynomial. After MULE computes all the roots of the polynomial a check is made for negligible roots. If the absolute value of any of these roots is less than RZERO (preset to 1.E-5) they will be set to zero. There is no specific ordering of the roots determined by MULE except that the complex roots are found in pairs. However, since a format for ordering these roots is defined in Section C.7 to provide a systematic development of routines utilizing root arrays, the routine RCLAS is called to reorder or to classify these roots. When the roots of a polynomial are found, the order in which they are listed in the printout will generally not be in the same order in which they are found.

LCAP2 commands which uses subroutine PROOT include PRTS, SPRTS, WPRTS, ZPRTS, SWXFM, SZXFM, SWMRX, SZMRX, and STIME.

C.1.2 MROOT1

Subroutine MROOT1 is used to compute the determinant of matrix M(s) in which the elements are polynomials. This matrix is represented in LCAP2 by

$$\mathbf{M}(s) = M4s^4 + M3s^3 + M2s^2 + M1s^1 + M0 \tag{C.3}$$

where M0, M1, M2, M3, and M4 are defined in Section 2.6.

Since the number of roots of the determinant of M(s) cannot be determined by inspection, the argument in the call to subroutine MULE for specifying the maximum number of roots to be found is set equal to MXPDEG[1], the maximum number of roots that can be stored in LCAP2. If there are more than MXPDEG roots only the first MXPDEG will be found and an error message will be printed out. The roots found by MULE, as in PROOT, will be compared with RZERO (preset to 1.E-7) to determine if there are any negligible roots. Any roots whose absolute value is less than RZERO will be set to zero. Also, the roots will be reordered or reclassified by the routine RCLAS.

Finding the roots of the determinant does not complete the evaluation of the determinant of a polynomial matrix. The gain associated with the roots must be determined. Since the root form representation

$$GAIN \cdot s^{\mu} \sum_{i=1}^{n-\mu}\left(\frac{s}{-a_i} + 1\right) \tag{C.4}$$

where μ = number of roots at the origin, as defined in Section 6.1.2, is used by LCAP2 instead of (C.1), the gain GAIN, which is also the low order nonzero coefficient of the polynomial, is given by

$$GAIN = \frac{F(z)}{z^{\mu}\sum_{i=1}^{n-\mu}(-\frac{z}{z_i} + 1)}\Big|_{z \neq z_i \ (i=1,n)} \tag{C.5}$$

The z used to evaluate this expression is determined by initially using (0.5 + j0.) and, successively, if necessary, adding (.1 + j0.) to this value until a z is found which is different from any of the roots. Since both root and coefficient form representations of polynomials are used in LCAP2, after the roots of the determinant are found, the coefficients of the polynomials will be computed from the roots and the low order nonzero coefficient. In the printout of the determinant the roots of the determinant will be printed out first along with the low order nonzero coefficient. This will then be followed by the coefficients of the polynomial.

[1] MXPDEG cannot be changed by the user. A new object and binary library has to be created for each different value of MXPDEG.

This method of evaluating the determinant by finding the roots first as opposed to finding the coefficients of the polynomial first is inherently more accurate.

LCAP2 commands which uses subroutine MROOT1 are DETRM and DTERM.

C.2 Eigenvalues and Generalized Eigenvalues

Computation of eigenvalues and generalized eigenvalues are used by commands B1EIG, B1LOCI, B1TF, B2EIG, B2LOCI, and B2TF. For a system represented in state space form,

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{ax} + \mathbf{b}u \\ \mathbf{y} &= \mathbf{cx} + \mathbf{d}u \end{aligned} \tag{C.6}$$

the eigenvalues of the system are computed by solving for the nonzero solution of

$$\mathbf{a}\,\mathbf{x} = \lambda\mathbf{x} \tag{C.7}$$

or equivalently (with λ replaced by s),

$$det\,[s\mathbf{I} - \mathbf{a}] = \mathbf{0} \tag{C.8}$$

For SISO transfer function evaluation, the transfer function from input u to output y_j is given by,

$$\frac{y_j(s)}{u(s)} = \frac{\det\begin{bmatrix} s\mathbf{I} - \mathbf{a} & \mathbf{b} \\ \text{row j of } -\mathbf{c} & \text{row j of } \mathbf{d} \end{bmatrix}}{\det\begin{bmatrix} s\mathbf{I} - \mathbf{a} \end{bmatrix}} \tag{C.9}$$

Computation of the poles of the transfer function is identical to the eigenvalue problem of (C.8). The zeros of the the transfer function is computed by solving for the nonzero solution (generalized eigenvalues) of

$$\mathbf{Ax} = \lambda\mathbf{Bx} \tag{C.10}$$

where,

$$\mathbf{A} = \begin{bmatrix} \mathbf{a} & -\mathbf{b} \\ row\ j\ of\ \mathbf{c} & row\ j\ of - \mathbf{d} \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{I} & 0 \\ 0 & 0 \end{bmatrix}$$

or equivalently (with λ replaced by s),

$$det[s\mathbf{B} - \mathbf{A}] = \mathbf{0} \tag{C.11}$$

The eigenvalues of (C.8) are computed by the QR method using driver subroutine EIGVAL from NASA/GODDARD Space Flight Center's SAMSAN [10] library of code for control system analysis and simulation. This routine in turn calls subroutines BALANC, HQR, and ORTHES which are from EISPACK [11]. After the eigenvalues are computed, the "gain" associated with these eigenvalues is computed so that the coefficient form of the characteristic polynomial can be computed[1]. Next, subroutine QRDRTS is called to check if the absolute value of any of the eigenvalues is less than RZERO (preset to 1.E-5). If any are found, they are set to zero. Also checks are made on complex roots to determine if the ratios of either |real/imag| or |imag/real| are greater than ERCX (preset to 1.E4). If any are found, the negligible part is set to zero. The calls to subroutines EIGVAL and QRDRTS are in subroutine EIGQR.

The generalized eigenvalues of (C.11) are computed by the QZ method using driver subroutine EZGVAL from the SAMSAN library. This routine in turn calls subroutines BALGEN, QZHES, QZIT, and QZVAL. BALGEN is from the SAMSAN library while QZHES, QZIT, and QZVAL are from EISPACK. The QZ method does not compute the eigenvalues directly nor return the number of eigenvalues. It computes (from EZGVAL) an n element complex array ALPHA and an n element real array BETA, where n is equal to the order of matrices **A** and **B**. An eigenvalue is defined from the ratio

$$\text{ALPHA(i)/BETA(i) for BETA(i)} \neq 0.$$

For an infinite precision machine, the number of eigenvalues will be equal to n minus the number of BETA(i)'s which are equal to zero. In practice, the number of eigenvalues will be computed as n minus the number of "∞" eigenvalues, where "∞" is defined when

- BETA(i)=0.
- CABS(ALPHA(i)/BETA(i)).GT.RTMAX

This calculation is performed in subroutine QZDRTS. Like subroutine QRDRTS, this routine will also set negligible eigenvalues to zero if the absolute value is less than RZERO. It will also set negligible imaginary or real parts of complex eigenvalues to zero if either |real/imag| or |imag/real| are greater than ERCX. The calls to subroutines EZGVAL and QZDRTS are in subroutine EIGQZ.

C.3 Partial Fraction Expansion

The partial fraction expansion of a rational function is used in determining both the inverse Laplace transform and the sampled-data transform. The particular expansion used by LCAP2 requires that there be no nonzero multiple poles and the number of numerator roots does not exceed the number of nonzero denominator roots. For most physical systems these constraints can be easily met. If the system has multiple poles not at the origin, they may be represented by separate poles displaced

[1] See (C.4)

from each other by a slight amount without appreciably affecting the results. Most physical systems, when properly modeled, will not have more zeros than poles.

For a rational function satisfying the above constraints, the function can be expressed in factored form as

$$G(s) = K_{RL}\frac{\sum_{j=1}^{q}(s-a_j)}{s^{\mu}\sum_{i=1}^{p}(s-b_i)} \tag{C.12}$$

where,

q = number of numerator roots
μ − number of poles at the origin
p = number of nonzero poles
K$_{RL}$ = is the root locus gain

The partial fraction expansion of G(s) is given by

$$G(s) = \sum_{i=1}^{\mu}\frac{\lambda_i}{s^i} + \sum_{i=1}^{p}\frac{\gamma_i}{s-b_i} \tag{C.13}$$

where λ_i and γ_i are given by

$$\gamma_i = K_{RL}\frac{\sum_{j=1}^{q}(b_i-a_j)}{b_i^{\mu}\sum_{\substack{j=1\\ j\neq i}}^{p}(b_i-b_j)} \tag{C.14}$$

$$\lambda_i = K_{RL}\delta_{pq}\delta_{i\mu} - \sum_{j=1}^{p} b_j^{i-1}\gamma_j \quad 1 \leq i \leq \mu \tag{C.15}$$

and δ_{pq} is the Kroneker δ function.

C.4 Inverse Laplace Transformation and Time Response

The inverse Laplace transform is evaluated by the partial fraction expansion method. For rational functions satisfying the constraints stated in Section C.3, the Laplace transform can be expressed in partial fraction form as

$$G(s) = \sum_{i=1}^{\mu}\frac{\lambda_i}{s^i} + \sum_{i=1}^{p}\frac{\gamma_i}{s-b_i} \tag{C.16}$$

where,

μ = number of poles at the origin
p = number of nonzero poles
λ_i and γ_i are defined by (C.15) and (C.14).

Separating the last term of (C.16) into the contribution for complex and real roots, G(s) becomes,

$$G(s) = \sum_{i=1}^{\mu} \frac{\lambda_i}{s^i} + \sum_{i=1}^{r} \frac{\alpha_{1i}s + \alpha_{0i}}{s^2 + \beta_{1i}s + \beta_{0i}} + \sum_{i=2r+1}^{p} \frac{\gamma_i}{s - b_i} \qquad (C.17)$$

where,

r = number of pairs of complex poles
α_{0i} = -2[Re{γ_{2i-1}} Re{b_{2i-1}} + Im{γ_{2i-1}} Im{b_{2i-1}}]
α_{1i} = 2Re{γ_{2i-1}}
β_{0i} = $|b_{2i-1}|^2$
β_{1i} = -2Re{b_{2i-1}}

Inverse transforming (C.17) into the time domain,

$$g(t) = \sum_{i=1}^{\mu} \frac{\lambda_i}{(i-1)!} t^{i-1} + \sum_{i=1}^{r} [\alpha_{1i} cos\omega_i t + \frac{\rho_i}{\omega_i} sin\omega_i t] e^{-c_i t} + \sum_{i=2r+1}^{p} \gamma_i e^{b_i t} \qquad (C.18)$$

where,

$c_i = \beta_{1i}/2$
$\omega_i = [\beta_{0i} - c_i^2]^{1/2}$
$\rho_i = \alpha_{0i} - \alpha_{1i} c_i$

C.5 Single Rate Sampled-Data Transforms

The z plane and the w plane sampled-data transform of the system given by Figure C.1 is computed by the partial fraction method [12,13] to yield a rational function.

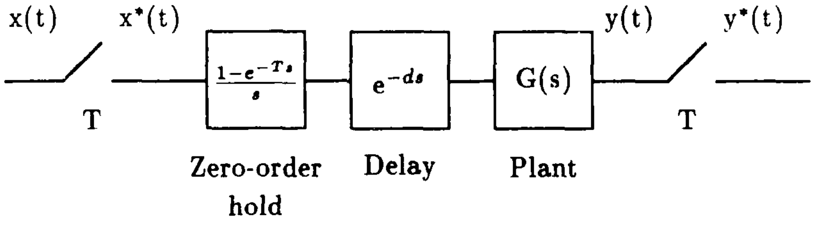


Figure C.1: Continuous System with Sampling

Using classical z transform theory, the simplest method for computing a z transform is to (1) perform a partial fraction expansion of the continuous system and (2) compute the z transform for each of the individual terms. In this unrationalized form, a z plane frequency response can be accurately evaluated. If only frequency responses are to be computed, this form of the z transform is sufficient. However, the rationalized form is usually desired since both the roots and the coefficients of the numerator and denominator are necessary for analysis using the block diagram reduction method. It is in rationalizing this sum of z transforms that the numerator roots and the coefficients of the transfer function will be subject to severe loss of accuracy when the order of the system is not small. This inherent inaccuracy, due to the mapping of all poles close to the origin in the s plane into the region of $z \cong 1$, can be minimized by performing all possible calculations in the w plane as defined by the bilinear transformation

$$w = \frac{z-1}{z+1} \tag{C.19}$$

Consider the system in Figure C.1. The transfer function between $x^*(t)$ and $y^*(t)$ is given by

$$H^*(s) = \frac{y^*(s)}{x^*(s)} = \frac{1}{2\pi j}\int_{c-j\infty}^{c+j\infty}\left[\left(\frac{1-e^{-T\xi}}{\xi}\right)e^{-d\xi}G(\xi)\right]\frac{d\xi}{1-e^{-T(s-\xi)}} \tag{C.20}$$

For the general case of time delays greater than one sampling period, it is convenient to define quantities k and Δ as follows:

$$d = (k-\Delta)T \quad \text{where } k \text{ is an integer such that } k-1 < d/T \leq k, \quad 0 \leq \Delta < 1 \tag{C.21}$$

Substituting (C.21) into (C.20) yields

$$H^*(s) = \frac{1}{2\pi j}\int_{c-j\infty}^{c+j\infty}\left(\frac{1-e^{-T\xi}}{e^{kT\xi}}\right)\left(\frac{G(\xi)}{\xi}e^{T\Delta\xi}\right)\frac{d\xi}{1-e^{-T(s-\xi)}} \tag{C.22}$$

Once again, restricting $G(s)$ as in (C.13), the partial fraction expansion of $G(\xi)/\xi$ can be expressed in partial fraction form and substituted into (C.22) to give

$$H^*(s) = \frac{1}{2\pi j}\int_{c-j\infty}^{c+j\infty}\left(\frac{1-e^{-T\xi}}{e^{kT\xi}}\right)\left(\sum_{i=1}^{\mu+1}\frac{\lambda_i}{\xi^i} + \sum_{i=1}^{p}\frac{\gamma_i}{\xi_i-\beta_i}\right)\frac{e^{T\Delta\xi}d\xi}{1-e^{-T(s-\xi)}} \tag{C.23}$$

Evaluating the above integral and letting $z = e^{Ts}$ and $\beta_i = e^{b_iT}$,

$$H(z) = \mathcal{Z}[H^*(s)] = \left(\frac{z-1}{z}\right)(z^{-k})\left[\sum_{i=1}^{\mu+1}\lambda_i\frac{z\acute{P}_i(z)}{z-1} + \sum_{i=1}^{p}\frac{\gamma_i\beta_i^{\Delta}z}{z-\beta_i}\right] \tag{C.24}$$

where

$$\begin{aligned}\acute{P}_1(z) &= 1\\ \acute{P}_2(z) &= T\left(\Delta + \frac{1}{z-1}\right)\end{aligned}$$

$$
\begin{aligned}
\acute{P}_3(z) &= \frac{T^2}{2!}\left[\Delta^2 + \frac{2\Delta + 1}{z - 1} + \frac{2}{(z-1)^2}\right] \\
&\vdots \\
\acute{P}_i(z) &= \frac{1}{(i-1)!}\frac{z-1}{z}\lim_{a\to 0}(-1)^{i-1}\frac{\partial^{i-1}}{\partial a^{i-1}}\frac{ze^{-a\Delta T}}{z - e^{-aT}}
\end{aligned}
$$

Simplifying the right side of (C.24) gives

$$
H(z) = z^{-k}\left[\sum_{i=1}^{\mu+1}\lambda_i \acute{P}_i(z) + \sum_{i=1}^{p}\gamma_i\beta_i^{\Delta}\frac{z-1}{z-\beta_i}\right] \tag{C.25}
$$

Transforming from the z to the w plane, using the bilinear transformation $z = (1+w)/(1-w)$ produces

$$
\begin{aligned}
F(w) &= \mathcal{W}\{H(z)\} \\
&= \left(\frac{1-w}{1+w}\right)^k\left[\sum_{i=1}^{\mu+1}\lambda_i P_i(w) + \sum_{i=1}^{p} 2\gamma_i\beta_i^{\Delta}\frac{w}{w(1+\beta_i) + (1-\beta_i)}\right]
\end{aligned} \tag{C.26}
$$

where

$$
\begin{aligned}
P_1(w) &= 1 \\
P_2(w) &= T\left(\Delta + \frac{1 - w}{2w}\right) \\
P_3(w) &= T^2\left(\frac{1 + (2\Delta+1)w + [2\Delta(\Delta-1)]w^2}{4w^2}\right) \\
&\vdots \\
P_i(w) &= \frac{1}{(i-1)!}\frac{2w}{1+w}\left\{\lim_{a\to 0}(-1)^{i-1}\frac{\partial^{i-1}}{\partial a^{i-1}}\left[\frac{e^{-a\Delta T}(1-w)}{w(1+e^{-aT}) + (1-e^{-aT})}\right]\right\}
\end{aligned}
$$

Separating the complex poles from the nonzero real poles in the second summation term of (C.26) and defining new parameters results in the final form

$$
F(w) = \left(\frac{1-w}{1+w}\right)^k\left\{\sum_{i=1}^{\mu+1}\lambda_i P_i(w) + w\left[\sum_{i=1}^{r}\frac{\phi_{1i}w + \phi_{2i}}{\phi_{3i}w^2 + \phi_{4i}w + \phi_{5i}} + \sum_{i=2r+1}^{p}\frac{\psi_{1i}}{\psi_{2i} + \psi_{3i}}\right]\right\} \tag{C.27}
$$

where r = number of complex root pairs and

$$
\begin{aligned}
\phi_{1i} &= 4Re\{\gamma_{2i-1}\beta^{\Delta}_{2i-1}(1+\beta^{*}_{2i-1})\} \\
\phi_{2i} &= 4Re\{\gamma_{2i-1}\beta^{\Delta}_{2i-1}(1-\beta^{*}_{2i-1})\} \\
\phi_{3i} &= 1+2Re\{\beta_{2i-1}\}+\mid\beta_{2i-1}\mid^{2} \\
\phi_{4i} &= 2(1-\mid\beta_{2i-1}\mid^{2}) \\
\phi_{5i} &= 1-2Re\{\beta_{2i-1}\}+\mid\beta_{2i-1}\mid^{2} \\
\psi_{1i} &= 2\gamma_{2i-1}\beta^{\Delta}_{2i-1} \\
\psi_{2i} &= 1+\beta_{2i-1} \\
\psi_{31} &= 1-\beta_{2i-1}
\end{aligned}
$$

$F(w)$ is now expressed as a sum of polynomials with real coefficients. These terms are then summed to yield a rationalized form of $F(w)$. Subroutine PROOT is then used to find the roots of the resulting numerator polynomial to yield the zeros of $F(W)$.

The w plane transform command SWXFM is implemented using the procedure described above. When the z plane transform command SZXFM is implemented, the same code is used to first compute a w transform. The w plane zeros and poles are then transformed to the z plane using the bilinear transform. The z plane coefficients are computed using the z plane roots. For higher order transfer functions the z plane coefficients cannot be accurately represented by a computer unless multi-length words are used[1]. A frequency response of a z plane transfer function computed directly from a SZXFM command will not be subject to the type of errors described above since it will use the roots instead of the coefficients of the transfer function in computing the response.

C.6 Multirate Sampled-Data Transforms

Three basic types of multirate sampled-data transforms are implemented by LCAP2. Each of these types will apply only for the case where the ratios of the sampling periods or sampling rates are related by an integer. First the notation used for representing multirate transforms will be presented. The z transform of the slowest sampler will be denoted by T so that the z transform of a continuous transform $G(s)$ is denoted by

$$\mathcal{Z}^{T}\{G(s)\} = G^{T}(z) \tag{C.28}$$

If G(s) is sampled at a rate n (an integer) times faster, the faster rate z transform of G(s) is denoted by

$$\mathcal{Z}^{T/n}\{G(s)\} = G^{T/n}(z_n) \tag{C.29}$$

where $z_n = (z)^{1/n}$ since $z = e^{sT}$ and $z_n = e^{sT/n}$.

[1] This is the reason why block diagram algebra with z plane transfer functions is subject to computational errors for higher order systems. Although multi-length words can be used to accurately represent coefficients to some predefined accuracy, it is not practical to develop software which will handle multi-length words greater than 2 words, i.e., DOUBLE PRECISION in FORTRAN.

The first type of multirate transform is the slow-to-fast multirate transform which describes the input-output relationship of the system in Figure C.2

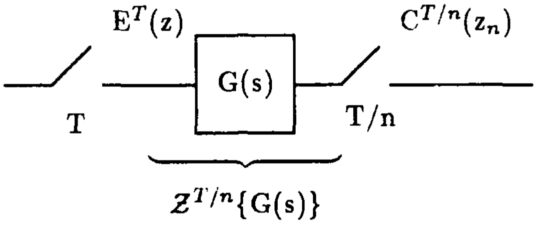


Figure C.2: Slow-to-Fast Multirate System

The output of this system is given by

$$C^{T/n}(z_n) = \mathcal{Z}^{T/n}\{G(s)\} * E^T(z) \tag{C.30}$$

If $G(s)$ is only a continuous function (i.e., no zero order hold) $\mathcal{Z}^{T/n}\{G(s)\}$ can be computed by the single rate z plane transform command SZXFM with the appropriate value for the sampling period. For multirate systems with a zero-order hold associated with $G(S)$, the period of the zero-order hold will be at the slower rate. Thus, if $G(s)$ is equal to a zero-order hold in cascade with $G_1(s)$, the transfer function between $E_T(z)$ to the output of $G_1(s)$ at the faster sampling rate is given by

$$G_1^{T/n}(z_n) = \mathcal{Z}^{T/n}\{\frac{1 - z^{-1}}{s} G_1(s)\} = \mathcal{Z}^{T/n}\{\frac{1 - z_n^{-n}}{s} G_1(s)\} \tag{C.31}$$

This transfer function can be computed in two steps by (1) using command SZXFM on the continuous function $G_1(s)/s$ with the sampling period set to T/n and (2) multiplying the results by $(1 - z_n^{-n})$. However, to simplify the number of operations for the user, the z plane multirate command SZMRX will perform these two steps automatically for the user. An equivalent w plane version of this command is SWMRX.

The second type of multirate transform is a z variable change command to allow a $G^T(z)$ transfer function at a slower rate to be changed to a faster rate representation, $G^{T/n}(z_n^n)$. The command which implements this transform is ZVCNG. See example in the Reference for ZVCNG in Appendix A.

The third type of multirate transform is the fast-to-slow multirate transform which describes the system shown in Figure C.3.

If $C^{T/n}(z_n)$ is resampled (skip sampled) at n times slower, the transform $C^T(z)$ is given by

$$C^T(z) = \frac{1}{n} \sum_{k=0}^{n-1} C^{T/n}(z_n e^{\frac{j2\pi*k}{n}}) \tag{C.32}$$

This multirate relationship is implemented in LCAP2 by two different methods. The first is a multirate frequency response method[1] which evaluates the response directly from (C.32) using

[1] Implemented by commands ZMRFQ and WMRFQ

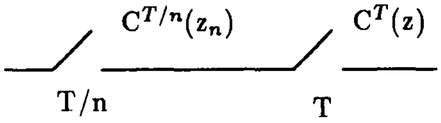


Figure C.3: Fast-to-Slow Multirate System

shifted values of z_n. This method will yield an accurate response but does not lend itself to a block diagram reduction analysis. The second method involves the evaluation of $C^T(z)$ in rational form as a function of z. The poles of $C^T(z)$ will be equal to the poles of $C^{T/n}(z_n)$ raised to the n-*th* power since $z = z_n^n$. The numerator roots of $C^T(z)$, however, are not simply related to the numerators of $C^{T/n}(z_n)$. At the present time, these numerator roots are solved for by casting the problem into a generalized root finding problem using the MULE root finding subroutine. This particular method for finding the numerator roots only works well for systems which are not too large. The accuracy in computing the rationalized form of (C.32) can be determined by evaluating the frequency response of the resulting transfer function and comparing it with the frequency response computed by using the ZMRFQ or WMRFQ commands.

C.7 Root Classification

In Chapter 6 the format used in LCAP2 for representing polynomials and transfer functions in root form was given. To summarize, (1) the real part of element one of a complex array is used to store the number of roots, (2) the imaginary part of element one of a complex array is used to store the nonzero coefficient of the polynomial, and (3) the roots are stored in consecutive elements of a complex array starting in element two. For the user, there is no constraint on the ordering of the roots when the data is entered except that a complex pair must be entered in consecutive locations. Once the data is entered, the roots are ordered so that a systematic development of routines utilizing root arrays can be implemented. The roots within the array are ordered as follows: complex, real (nonzero), and zero. After the user enters in a set of roots, the program will automatically order the roots into this format using subroutine RCLAS.

RCLAS is also used after roots and eigenvalues are found. In addition to ordering the roots, RCLAS will check the roots to (1) zero out any negligible real or imaginary parts of a root, (2) set any negligible roots to zero, and (3) print out an error message when complex roots do not exist in conjugate pairs. These operations on the roots tend to compensate for small errors introduced by the root finding routine by "forcing" roots to values which are more representative of physical systems. When computations such as partial fraction expansion and computation of polynomial coefficients from it factored form are performed, errors will be minimized when the roots have been "processed" by RCLAS.

The criteria for the operations described above are given below. For the root array x_i, i = 1,n execute steps 1 through 8 for each value of i.

1. Determination of Noncomplex Root
   If imag(x_i) = 0 the root is noncomplex, go to step 7; otherwise go to step 2.

2. Determination of Complex Root
   If $|\mathrm{real}(x_i)/\mathrm{imag}(x_i)| \leq$ ERCX (preset to 1.E4), the root is complex, go to step 4; otherwise go to step 3.

3. Set Negligible Imaginary Part to Zero
   $x_i = \mathrm{real}(x_i) + j0$, go to step 7.

4. Determination of Root on the Imaginary Axis
   If $\mathrm{real}(x_i) = 0$, go to step 8; otherwise go to step 5.

5. Determination If Real Part Is Negligible
   If $|\mathrm{imag}(x_i)/\mathrm{real}(x_i)| >$ ERCX, the real part is negligible, go to step 6; otherwise go to step 8.

6. Set Negligible Real Part to Zero
   $x_i = 0 + j\ \mathrm{imag}(x_i)$, go to step 8.

7. Move Noncomplex Roots to the End of the Array
   temp $= x_i$
   $x_j = x_{j+1}$, j=1,n-1 where n = number of roots in the array
   $x_n =$ temp

8. Continue

   For the complex roots found in steps 1 through 7, which are now in the first part of the array, execute steps 9 and 10 for i = 1,3,...,(n_c-1) where n_c is the number of complex roots.

9. Verify that x_i and x_{i+1} are Conjugate Pairs
   For i odd, if $\left| \frac{real(x_i)-real(x_{i+1})}{x_i} \right| <$ ERCNJ and if $\left| \frac{imag(x_i)+imag(x_{i+1})}{x_i} \right| <$ ERCNJ,
   x_i and x_{i+1} are conjugate pairs, go to step 11; otherwise go to step 10.
   (ERCNJ is preset to 1.E-4).

10. Search for Conjugate of x_i
    if an x_j can be found such that
    $\left| \frac{real(x_i)-real(x_j)}{x_i} \right| <$ ERCNJ and $\left| \frac{imag(x_i)+imag(x_j)}{x_i} \right| <$ ERCNJ
    where $i < j < n_c$, interchange x_j with x_{i+1}. If the conjugate of x_i is not found an error message will be printed out. Since an error in the complex roots will not necessarily result in erroneous results for all subsequent LCAP2 operations, the program is allowed to continue. The user should check to determine if the missing complex conjugate root(s) is due to an input error in loading roots.

11. For the remaining noncomplex roots the roots at the origin are found and moved to the end of the array. For i = (n_c+1), n execute steps 12 through 14.

12. Determine Roots at the Origin
    If $|x_i| <$ ERCZ (preset=1.E-5), the root is zero or negligible, go to step 13; otherwise go to step 14.

13. Move Roots at the Origin to the End of the Array
    $x_j = x_{j+1}$, j = 1,n-1
    $x_n = (0.,0.)$

14. Continue

C.8 Common Root Elimination

In the analysis of systems, quite often common roots occur between the numerator and denominator of transfer functions, particularly when transfer functions are evaluated by Cramer's method. Since these common roots are generally not identically equal, subsequent computations using the roots of the transfer function without first eliminating them will introduce errors. Subroutine CRELIM is used to determine and eliminate common roots from transfer functions. The criteria for common roots are:

If a numerator root a_i and a denominator root b_j of a transfer function are found such that,

$$\mid \frac{b_j}{a_i} - (1 + j0) \mid < ECRE1 \ for \ a_i \neq 0 \tag{C.33}$$

or

$$\mid b_j \mid < ECRE2 \ for \ a_i = 0 \tag{C.34}$$

a_i and b_j are considered to be common roots and will be eliminated from the transfer function. ECRE1 is defined as the CRELIM nonzero common root criterion and ECRE2 is defined as the CRELIM zero root criterion. The preset values for these parameters are:

ECRE1 = 2.E-4
ECRE2 = 1.E-8

The user, however, can change these values.

Appendix D

Changes Affecting Users of Previous Versions of LCAP2

In developing this version of LCAP2 emphasis was placed on upward compatibility with the previous version of the program. The names and usage of the LCAP2 commands and parameters were retained as much as possible. However, several factors influenced the decision to make some fundamental changes to the program. They included (1) code conversion from FORTRAN 66 (CDC FTN4) to FORTRAN 77 (CDC FTN5 and CRAY CFT), (2) renaming or redefining of some LCAP2 commands and parameters for more consistent usage, and (3) portability of LCAP2 source code for different size versions of the program.

D.1 Changes Related to Conversion from FORTRAN 66 to 77

In the previous version of the program, text data, such as labeling of plot titles, was done with the use of REAL variables which were set equal to Hollerith characters. This was not a very convenient method for handling text data, but that was all that was available with FORTRAN 66. Since the FORTRAN 77 language defines CHARACTER variables for handling text data, all REAL variables for text data in the old version were changed to CHARACTER variables where feasible.

In the old version the labeled common block,

```
      COMMON/HEADDB/HEAD(70),DB(900)
```

was replaced by four common blocks, HEAD, DBASE, POLYCM, and ROOTCM.

For CDC, common block HEAD is,

```
      COMMON/HEAD/HEAD
      CHARACTER*70 HEAD(5),TITLE1,TITLE2,TITLE3,TITLE4,SAVLBL
      EQUIVALENCE (HEAD(1),TITLE1),(HEAD(2),TITLE2),(HEAD(3),TITLE3)
     +,(HEAD(4),TITLE4),(HEAD(5),SAVLBL)
```

For CRAY, common block HEAD is,

```
      COMMON/HEAD/HEAD
      CHARACTER*64 HEAD(5),TITLE1,TITLE2,TITLE3,TITLE4,SAVLBL
      EQUIVALENCE (HEAD(1),TITLE1),(HEAD(2),TITLE2),(HEAD(3),TITLE3)
     +,(HEAD(4),TITLE4),(HEAD(5),SAVLBL)
```

For both CDC and CRAY, the common block DBASE is,

```
      COMMON/DBASE/DB(300)
```

For both CDC and CRAY, the common block POLYCM is,

```
      DIMENSION POLYP(MXPDEG+2),POLYN(MXPDEG+2),POLYD(MXPDEG+2)
      COMMON/POLYCM/POLYP,POLYN,POLYD
```

For both CDC and CRAY, the common block ROOTCM is,

```
      COMPLEX ROOTP(MXPDEG+1),ROOTN(MXPDEG+1),ROOTD(MXPDEG+1)
      COMMON/ROOTCM/ROOTP,ROOTN,ROOTD
```

Common block HEADDB was replaced by four common blocks for the following reasons:

- Common block HEAD for CHARACTER variables had to be defined separately since CHARACTER and REAL variables cannot be mixed in the same common block.
- REAL variables equivalenced to array DB were unchanged if they are not a function of the FORTRAN parameter MXPDEG, which is the maximum polynomial degree for a particular version of LCAP2.
- Arrays POLYP, POLYN, and POLYD used to enter polynomial coefficient data, which was in common block HEADDB, is now defined in common block POLYCM since their dimensions are a function of the FORTRAN parameter MXPDEG.
- Arrays ROOTP, ROOTN, and ROOTD used to enter polynomial root data, which was in common block HEADDB, is now defined in common block ROOTCM since their dimensions are a function of the FORTRAN parameter MXPDEG.

For labeling both low resolution printer plots and the high resolution electrostatic plots, the CHARACTER variable TITLE1 is used for the title of the plot, i.e.,

```
      TITLE1='EXAMPLE 1 S PLANE FREQUENCY RESPONSE'
```

For labeling the high resolution plots, the following CHARACTER variables, TITLE2, TITLE3, and TITLE4, can be used for additional lines of annotation on lines 2, 3, and 4, respectively.

The subroutines HEADINi, (i=1,5), used in the previous versions of LCAP2 for labeling plots are no longer available.

The array DB is used to define memory allocation for LCAP2 parameters. In routines which use these parameters, appropriate equivalence statements must be declared. Even though the dimension of DB has been reduced to 300 in common block DBASE, only five array parameters were affected by the conversion to FORTRAN 77. They are the arrays FREQi, i=1,5. They were formerly equivalenced to DB(i), i=698, 701, 704, 707, 710. They should now be equivalenced to DB(i), i=1, 4, 7, 10, 13.

D.2 Renaming of LCAP2 Commands

Changes were made to the following LCAP2 commands. The changes were either to the name or else to the number of arguments. Only the FORTRAN form (not its PRECMP form) is given below.

- RESTORE(IPRNFLG) → LOAD('file_name',iprn)
- STORE(IPRNFLG) → SAVE('file_name',iprn)
- WMRFQ(I,M) → WMFRQ(i)
- ZMRFQ(I,M) → ZMFRQ(i)

D.3 Renaming of LCAP2 Parameters

The following LCAP2 parameters were renamed. In most cases the new name will either be easier to remember or else better reflect the function that it represents.

- FDLAY → FDELAY
- FILM → HRDCPY
- FXYDL → FXYDEL
- FXYMN → FXYMIN
- MNDW → MINDW
- MXDW → MAXDW
- MXITF → MAXITF
- NOMEG → NOMEGA
- NRMFG → NRMHI
- POLY → POLYP
- ROOT → ROOTP
- RLXMN → RLXMIN

- RLXMX → RLXMAX
- RLYMN → RLYMIN
- RLYMX → RLYMAX
- TSTEP → TTYPE

D.4 Changes to Deck Setup for Creating the Main Program Using UPDATE

In the old version the main program was created for the user with the following UPDATE directives:

```
*IDENT identifier
*INSERT START.1
*DECK MAIN
*CALL LCAP2
      CALL INITO
      CALL MINITO
          ⋮
```

In the new version the following UPDATE directives are used,

```
*IDENT identifier
*INSERT START.1
     PROGRAM LCAP2(INPUT,OUPUT,TAPE5=INPUT,TAPE6=OUTPUT)
*CALL COMLCAP2
     CALL INITO
     CALL MINITO
          ⋮
```

In the deck setup for the old version of the program, COMDECK LCAP2 includes the program statement and many lines of common block and equivalence statements used for creating the first part of the users main program. For users who are not familar with UPDATE, this setup was probably confusing since the COMDECK and the main program being created were both called LCAP2. For the new version of LCAP2, the user must explicitly define the program statement. The *CALL COMLCAP2 directive will insert many different sets of common blocks and equivalence statements after the program statement. Whereas the common block and equivalence statements in COMDECK LCAP2 in the old version were actual source code, the statements in COMDECK COMLCAP2 for use in the new version are *CALL statements to other COMDECKs which contain the actual source code for the common block and equivalence statements. In addition to simplifying the maintenance of libraries of common decks used by LCAP2, the *CALL directives in COMDECK COMLCAP2 should encourage the user to make use of several COMDECKs when user subroutines are being written for more complex analysis. COMDECK COMLCAP2 consist of the following *CALL directives:

```
*CALL SETMXP
*CALL FRQBLK
*CALL PLOT1
*CALL FUNCTL
*CALL TERMNL
*CALL TFPCNT
*CALL TFTEMP
*CALL DBASE
*CALL DBEQUIV
*CALL HEAD
*CALL POLYCM
*CALL ROOTCM
*CALL INTCOM
*CALL MATRIX1
*CALL MDET1
*CALL SCMBLK
*CALL CMPOLY
*CALL SETB1
*CALL SETB2
*CALL IYCDS
```

When a user writes his or her own subroutines which use LCAP2 parameters, (1) the common blocks HEAD, DBASE, POLYCM, and ROOTCM must be declared and (2) LCAP2 parameters which are not in common blocks POLYCM and ROOTCM must be equivalenced to the appropriate elements of the array DB in common block DBASE. This can be easily be done with the use of the following UPDATE directives:

```
*CALL SETMXP
*CALL DBASE
*CALL DBEQUIV
*CALL HEAD
*CALL POLYCM
*CALL ROOTCM
```

The *CALL SETMXP directive will insert the FORTRAN PARAMETER statement declaring the value of MXPDEG.

Appendix E

Notes on Creating Plots

Both low resolution printer plots and high resolution electrostatic plots can be generated whenever a frequency response, time response, or root locus command is invoked. The low resolution plot is created if the value of the LCAP2 parameter GRAFP is nonzero. The high resolution plot is created if the value of the LCAP2 parameter HRDCPY is a nonzero. The following are notes on creating plots.

E.1 Plot Labels

The CHARACTER variables TITLE1, TITLE2, TITLE3, and TITLE4 in common block HEAD are used for labeling plots. Each of these variables corresponds to one line of annotation. TITLE1 is used by both the low resolution and high resolution plots. TITLE2, TITLE3, and TITLE4 are used only by the high resolution plots and are not available for low resolution plots. TITLE1 is always positioned at the top of a plot. Title data is entered using a character expression, i.e.,

```
          TITLE1='FREQ. RESPONSE FOR FLEX MODE=  1.20, ZETA= .005'
```

Character substring reference can be used to change part of a CHARACTER variable. For example, to change the value of flex mode and zeta in variable TITLE1 above, to 4.50 and .008, respectively, the following statements can be used,

```
          TITLE(30:35)='  4.50'
          TITLE(43:47)=' .008'
```

Another method of changing a substring of a CHARACTER variable is to use an internal write statement. For example, if five different frequency response plots are to be made with different sets of flexible mode frequency and damping coefficients, the following code can be used to automatically annotate the plot title:

```
          TITLE1='FREQ. RESPONSE FOR FLEX MODE=      , ZETA=     '
          DO 100 I=1,5
    C           .
```

```
      WFLEX = ...
      WZETA = ...
C       .
C     "compute transfer function and store in SPTF10"
      WRITE(TITLE1(30:35),'(F6.2)')WFLEX
      WRITE(TITLE1(43:37),'(F5.3)')WZETA
C       .
C     "enter frequency response parameters"
C       .
  100 CALL SFREQ(10)
```

For high resolution plots, additional lines of annotation are entered with the variables TITLE2, TITLE3, and TITLE4. The actual number of lines generated on a plot is determined by the parameter NANOT (preset to 4) which can be 0 - 4. This parameter was defined to allow the user to suppress plot annotation without having to set a title variable to a blank word[1].

The second, third, and fourth line of annotation normally begins just below the title at the top of the plot. The start of the second line of annotation is controlled by the value of parameter YANOT (preset to 9.6 inches). If pertinent plot data is located near the top of the plot, the additional lines of annotation can be moved to the lower part of the plot be setting YANOT to a smaller value (range = 0-10 inches).

E.2 Low Resolution Printer Plots

The default settings for the size of the printer plot is 52 rows high and 103 characters wide. The parameter which controls the height is XLINES. The parameter which controls the width is XNCOL which can be either 73 or 103. (The printer plots for the examples in Chapter 8 were generated using XLINES=43 and XNCOL=73.) Multiple plots (overlays) cannot be produced on printer plots.

E.3 High Resolution Plots

The size of the high resolution plots cannot be changed by the user. The low level Aerospace plot routines are used to create the plots in LCAP2. The printer units used by the Aerospace plot routines are in inches when used with a pen plotter. When used with a Versatec electrostatic printer, a nominal 10x17 inch size pen plot is scaled to fit on an 8 1/2 x 11 paper. An inch in printer units on a electrostatic plot is thus approximately 0.59 inches.

E.3.1 Multiple Plots

The default mode for plotting the high resolution plots is one curve per plot. For commands which generate only one plot per command[2], the frame advance for plots can be inhibited so that multiple

[1] A CHARACTER variable can be set to a blank string as follows, i.e., TITLE4=' '

[2] Examples are: (1) STIME, (2) ZTIME, or (3) any frequency response command with only the FNICO or FNYQS option. The FBODE option cannot be used for multiple plots since it will create two plots per command.

curves can be produced on a single plot. Thus, to plot n curves on a plot, n commands must be specified with the appropriate parameter for controlling the frame advance. This parameter is CONTP which is preset to 0. This parameter is defined as:

CONTP = 0	Single curve plot
CONTP = 1	First curve of a plot
CONTP = 2	Continuation of a plot
CONTP = 3	Final curve of a plot

When a plot with multiple curves is to be generated, the first command must be made with CONTP=1. This will inhibit the frame advance in addition to creating the grid, axis, annotations, and the first curve. For subsequent curves, but not the final curve, the value of CONTP must be set to 2. The final curve is to be made with CONTP=3 which will reset the inhibit of the frame advance. An example for creating a multiple plot for frequency responses contained in **SPTF$_4$**, **SPTF$_5$**, **SPTF$_6$**, **SPTF$_7$**, and **SPTF$_8$** is given below:

```
    FBODE=0                   "no Bode plot"
    FNYQS=0                   "no Nyquist plot"
    FNICO=1                   "only Nichols plot"
    DO 100 I=4,8
    CONTP=2
    IF(I.EQ.4)CONTP=1
    IF(I.EQ.8)CONTP=3
100 CALL SFREQ(I)
```

E.3.2 Intensity and Zero Line

The intensity of a curve drawn on a plot can be controlled by the parameter SHADE which is defined from 2 to 28, with 28 being the darkest. The preset value of SHADE is 16. The intensity level can be changed when making multiple curve plots so that the curves can be differentiated.

To improve the appearance of plots there is a provision to darken the grid lines when the horizontal or vertical scale changes sign. The intensity for the zero line is specified by the parameter ZLINE (preset to 19).

E.3.3 "Pen" Pickup

The high resolution frequency and time response plots are created by connecting a series of response points by straight lines. Even though the responses may be evaluated with sufficiently small increments to allow a smooth curve to be plotted, the user's choice for the plot scale or the nature of the plot may still produce discontinuities. The most obvious example is the Nichols plot in which the phase is cyclic. To avoid a wraparound the "pen" must be picked up when it reaches one end of the axis and continues on the opposite end. The parameters XGAP and YGAP are used to determine when points are not to be connected. When the delta change in either the x or y direction of the data to be plotted exceeds XGAP or YGAP (in 0.01 printer inch units) the points are not connected. XGAP and YGAP are preset to 500[1] which corresponds to one half of the plot length in the y direction. When points are not connected, the points are marked with a small box.

[1] In the previous versions of LCAP2, XGAP and YGAP were preset to 50. This caused discontinuous curves to be drawn when the points were more than 0.5 printer inch units apart.

Appendix F

Line Printer Output Suppression

There is a provision to suppress all or a portion of the output generated by the line printer. This feature can be used to eliminate the printout of intermediate results once the program has been checked out. Flags PRNFLGi, i=1,5 in common block PRNCTL are defined for suppressing the output. At the present time only PRNFLG2, PRNFLG3, and PRNFLG4 (all preset to 1) are used. The printer output from an LCAP2 command generally can be separated into three parts, (1) input polynomial or transfer function arguments, (2) summary of the LCAP2 command, and (3) output polynomial or transfer function computed or computation of frequency response, time response, or root locus. The printer output for these three parts can be controlled by PRNFLG2, PRNFLG3, and PRNFLG4, respectively. The output is suppressed if these parameters are set to 0. These flags can be changed during a run so that selected output can be generated.

As as example, the command "**CALL PLDC**" in Example 2 in Chapter 8 generated the following output:

```
DEGREE OF POLYP   IS  3          (COEFFICIENTS IN ASCENDING ORDER)
34. 38. 13. 1.

************************************************************
*     PLDC - POLYNOMIAL LOAD IN COEFFICIENT FORM           *
************************************************************

DEGREE OF POLY1   IS  3          (COEFFICIENTS IN ASCENDING ORDER)
34. 38. 13. 1.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

If PRNFLG2=0 polynomial POLYP would not have been printed out. If PRNFLG3=0, both (1) the summary of the PLDC command, enclosed by the asterisk box, and (2) the dashed line designating the end of a command, would not have been printed out. If PRNFLG4=0, the polynomial POLY1 would not have been printed out.

Appendix G

Reserved Files for LCAP2

This appendix lists the files used by LCAP2. These files should not be used by the typical user when developing an LCAP2 analysis program.

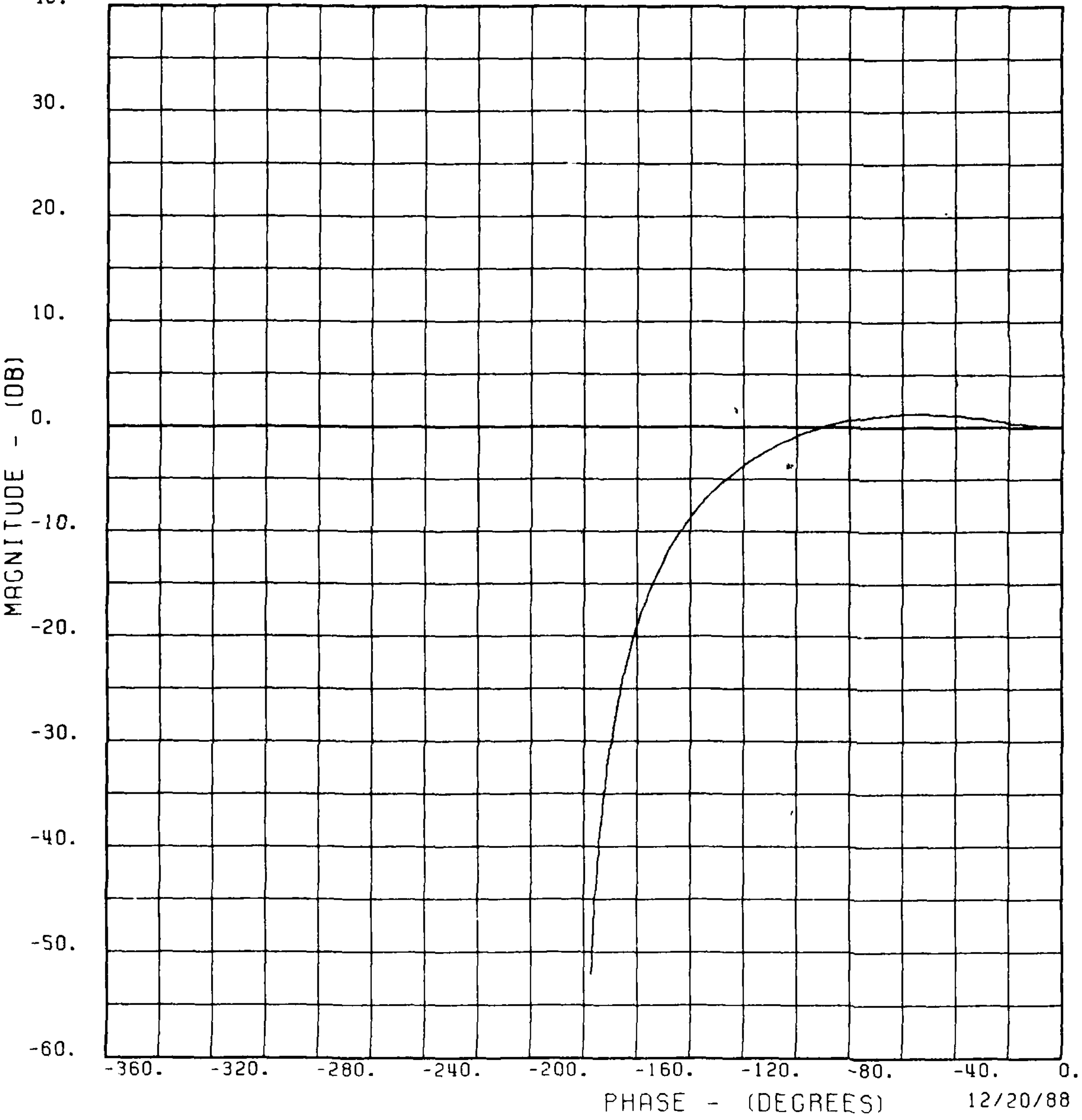
Table G.1: Reserved Files For LCAP2

tape_name	Description
TAPE19	Scratch file for storing roots computed by B1LOCI and B2LOCI
TAPE29	Scratch file used by interactive command CDCFILE
TAPE30	Scratch file used by command LOAD
TAPE31	Scratch file used by command SAVE
TAPE32	Scratch file used by command B1LOAD
TAPE33	Scratch file used by command B1SAVE
TAPE35	Scratch file used by commands B1SAVE and B1LOAD
TAPE50	File used by CDC PRECMP
TAPE83	File used for saving matrix data in interactive LCAP2
TAPE84	Direct access file for storing $\mathbf{POLY}_i$ arrays
TAPE85	Direct access file for storing $\mathbf{SPTF}_i$ arrays
TAPE86	Direct access file for storing $\mathbf{WPTF}_i$ arrays
TAPE87	Direct access file for storing $\mathbf{ZPTF}_i$ arrays
TAPE89	File for storing counter NPLOTS in interactive LCAP2

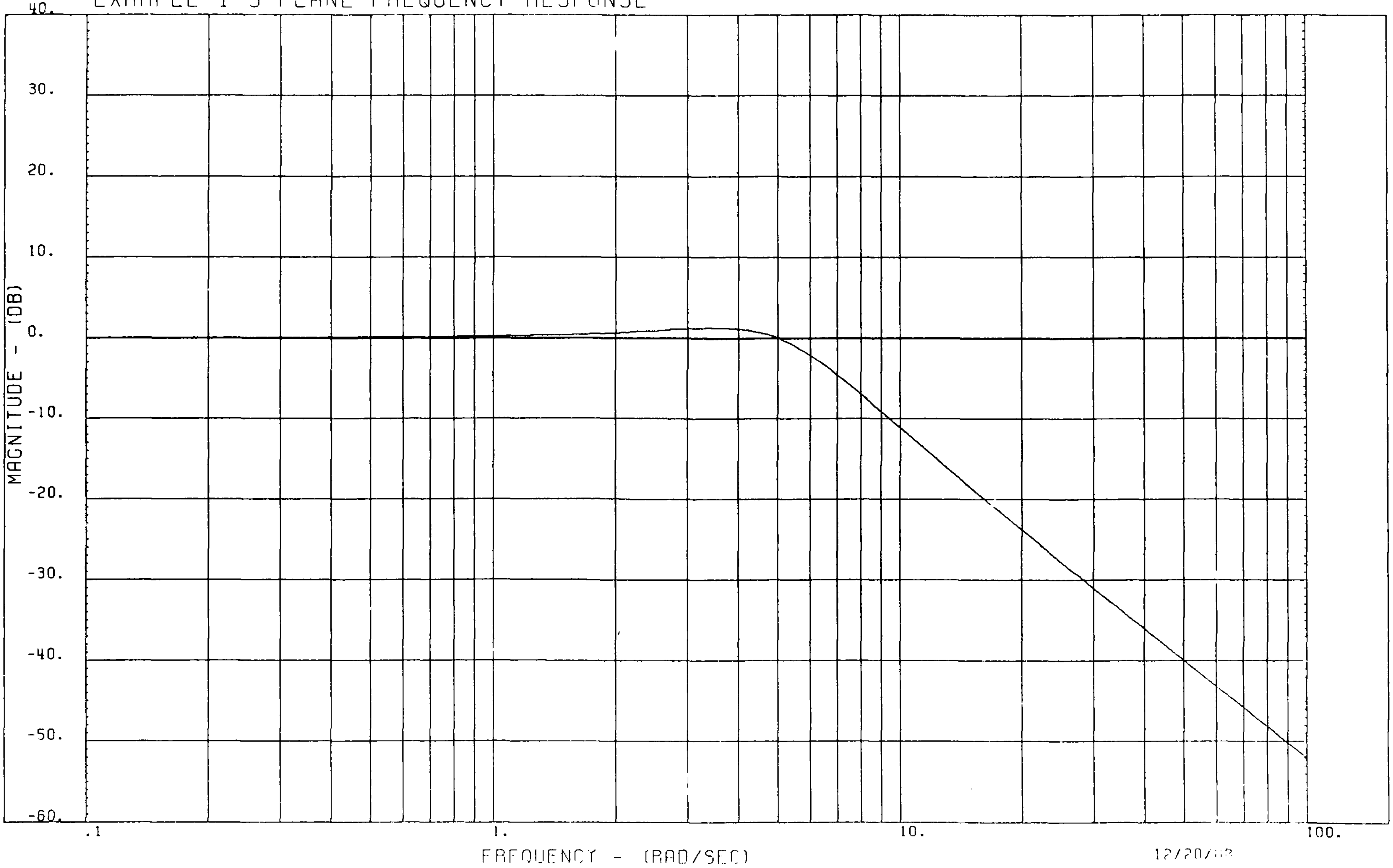
Appendix H

Hardcopy Plots for the Examples from Chapters 8 and 9

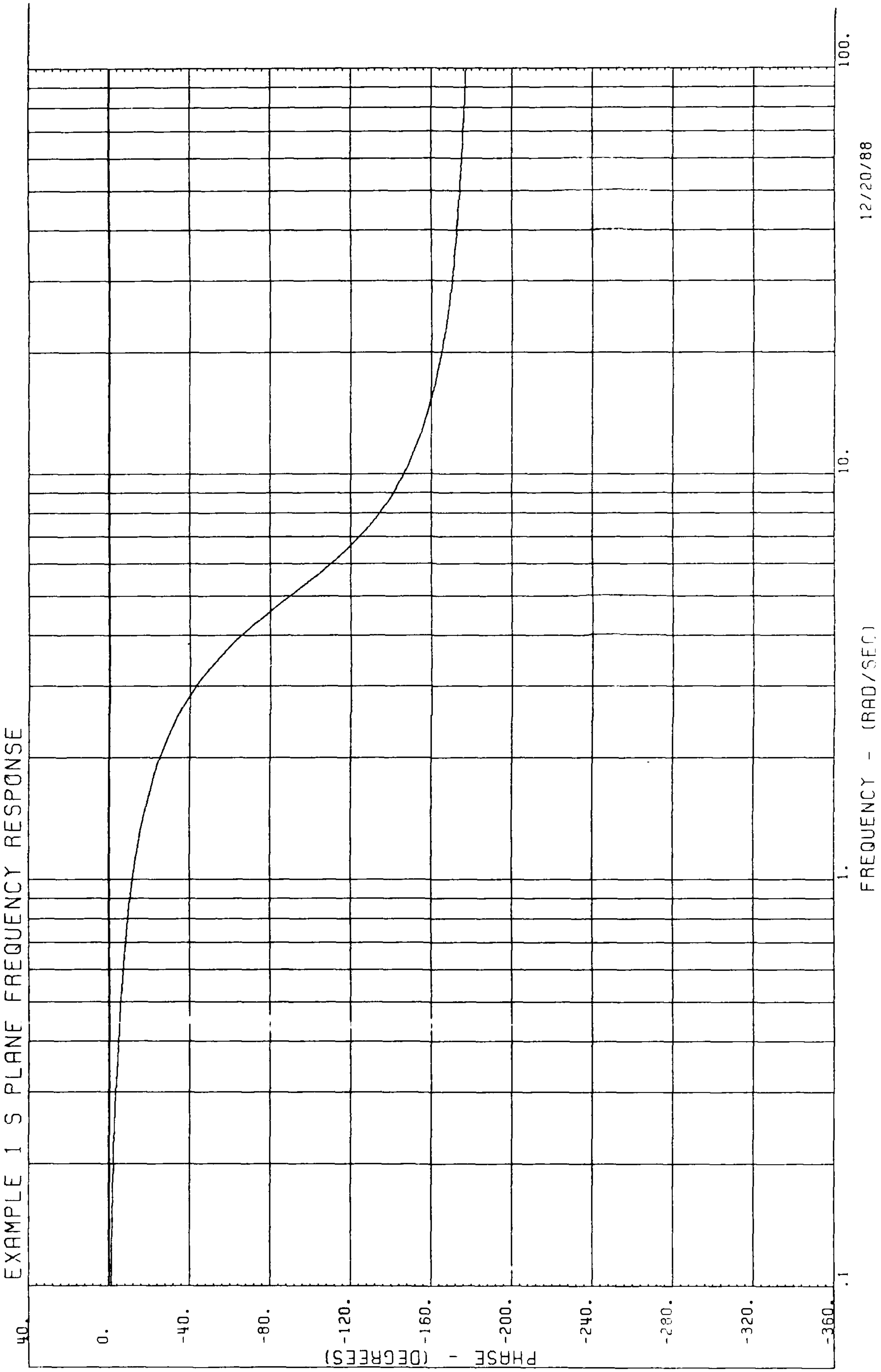
EXAMPLE 1 S PLANE FREQUENCY RESPONSE



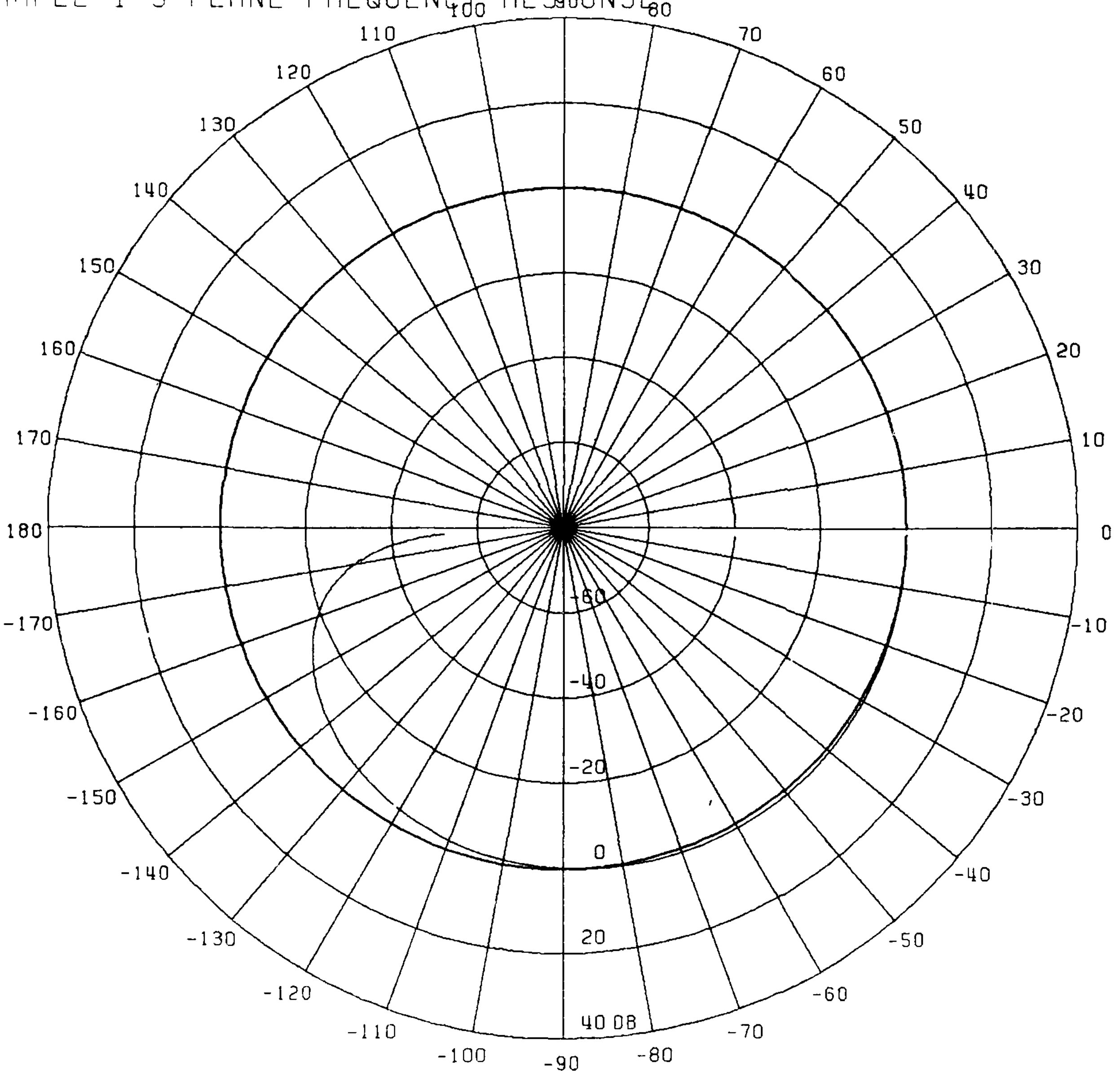
EXAMPLE 1 S PLANE FREQUENCY RESPONSE



12/20/88

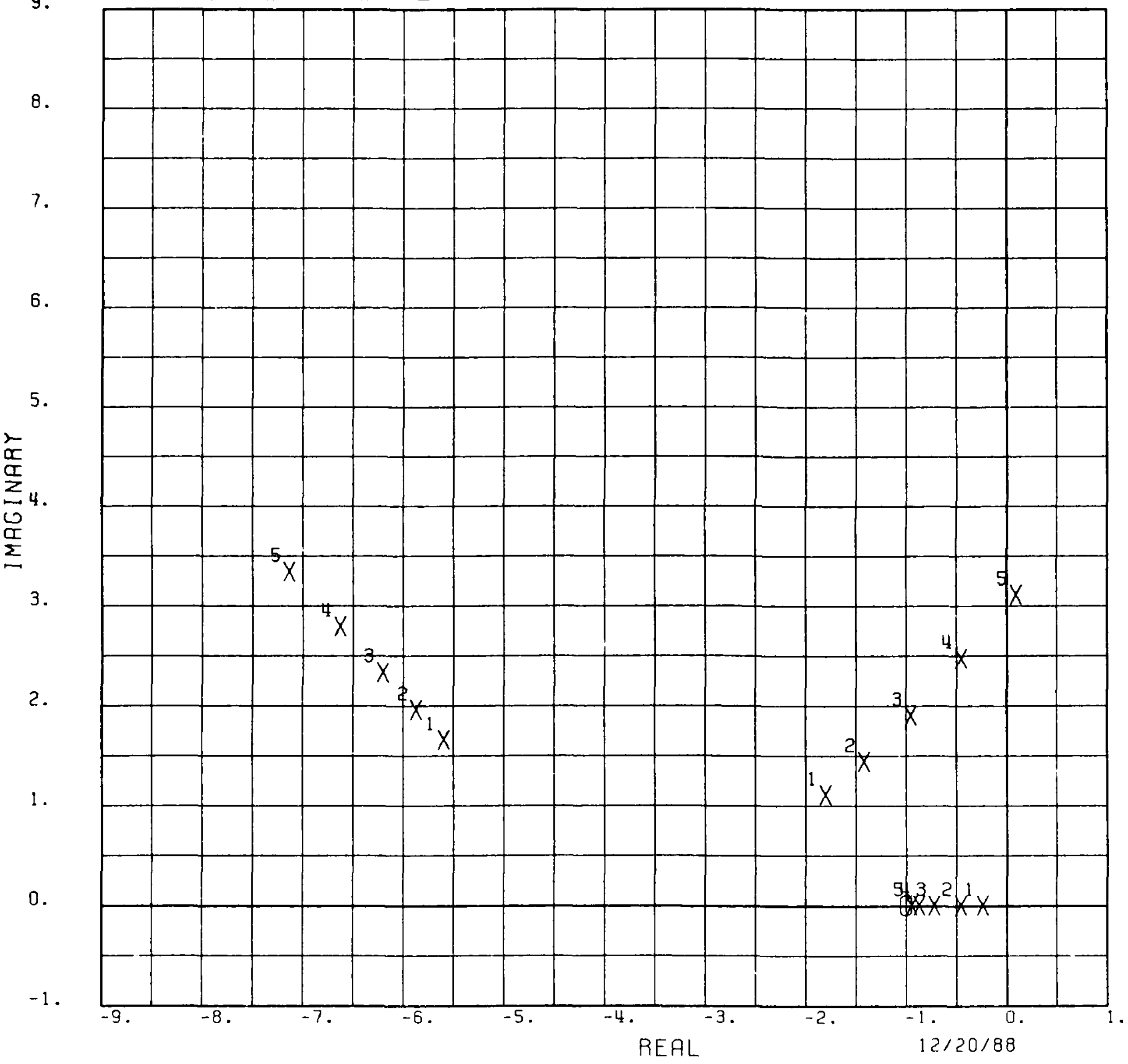

EXAMPLE 1 S PLANE FREQUENCY RESPONSE
12/20/88
FREQUENCY - (RAD/SEC)
.1
1.
10.
100.
PHASE - (DEGREES)
40.
0.
-40.
-80.
-120.
-160.
-200.
-240.
-280.
-320.
-360.

EXAMPLE 1 S PLANE FREQUENCY RESPONSE

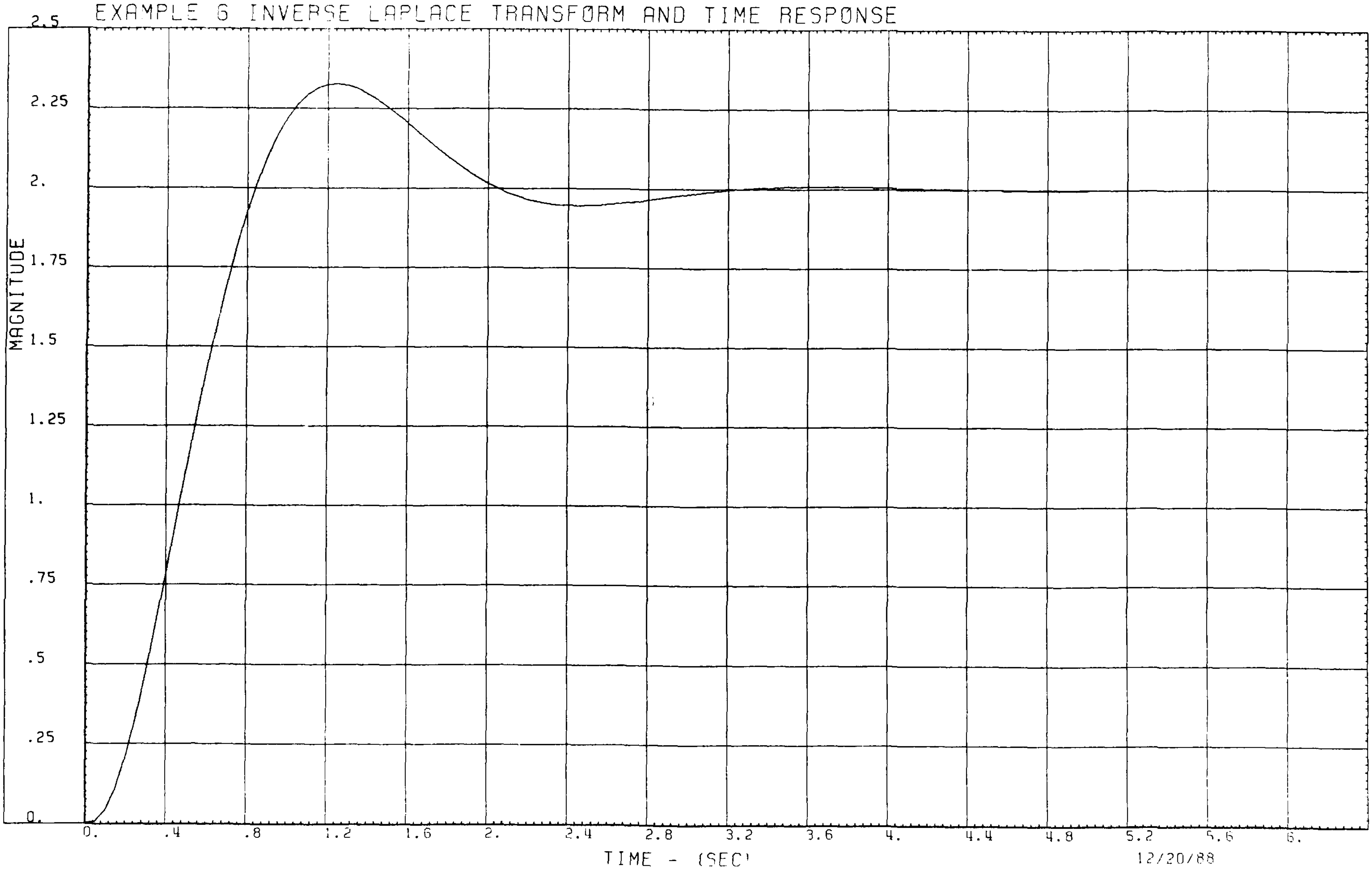


12/20/88

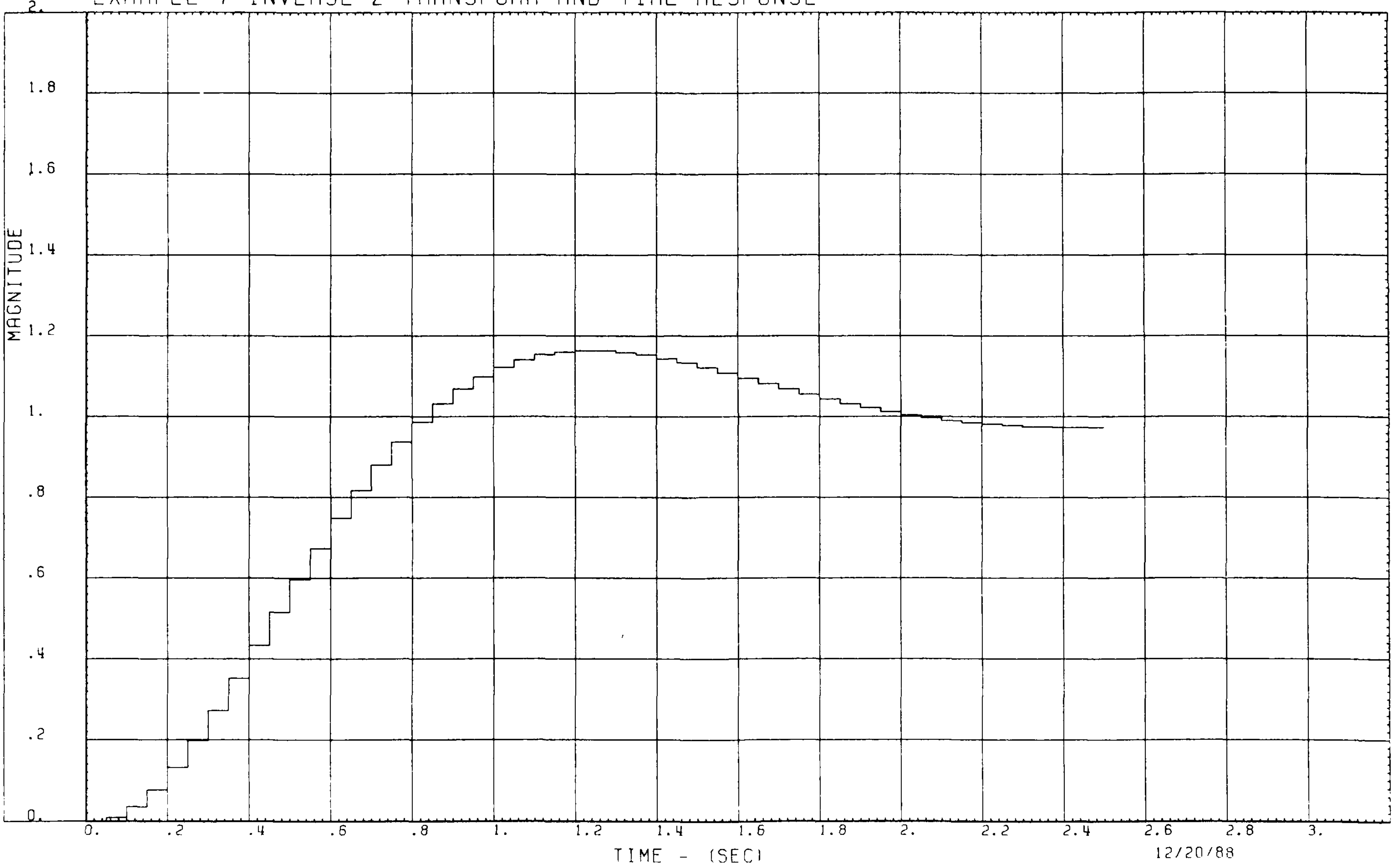
EXAMPLE 5 S PLANE ROOT LOCUS



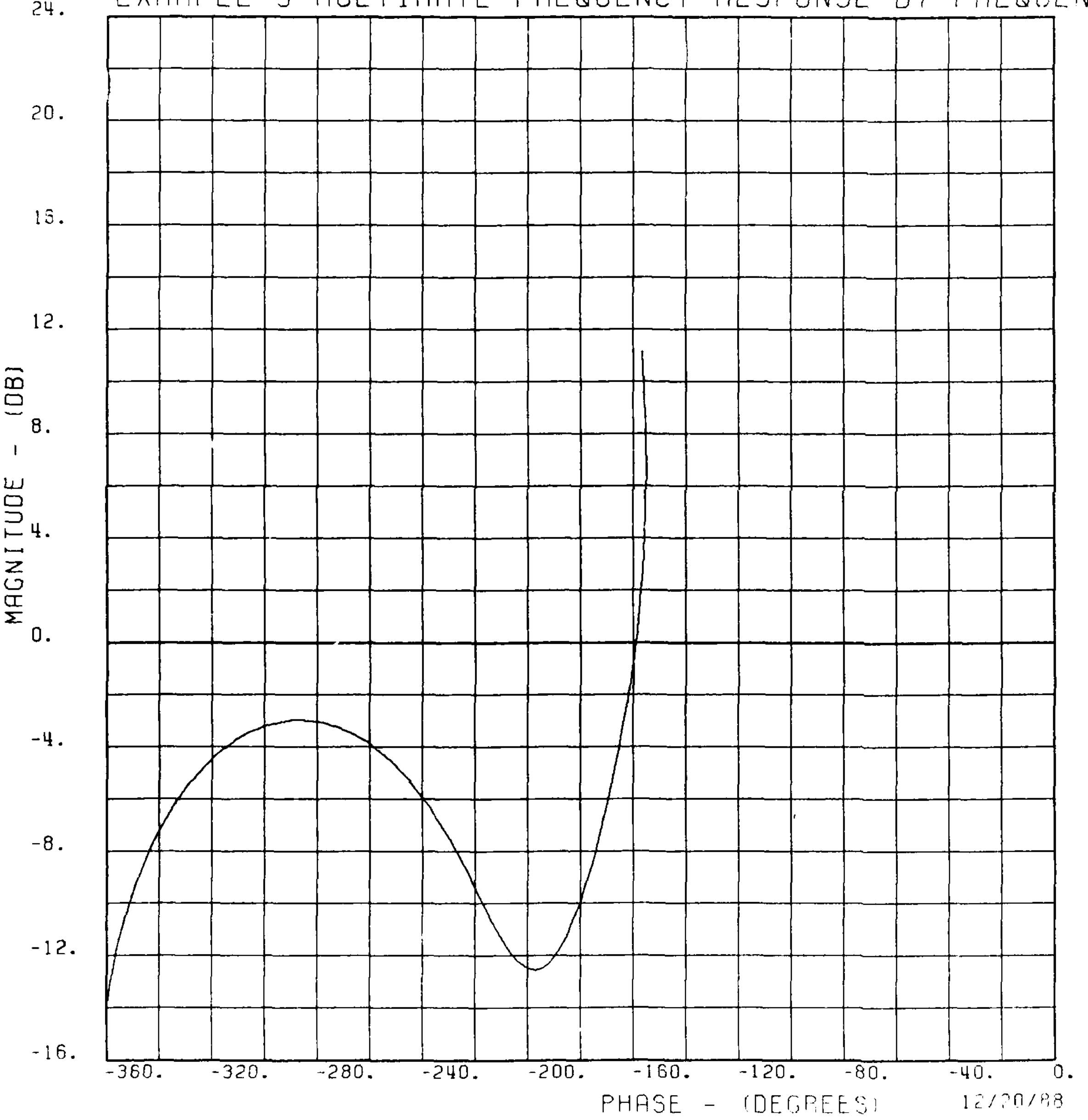
GAIN NO.	GAIN
1	.12500000
2	.25000000
3	.50000000
4	1.0000000
5	2.0000000


EXAMPLE 6 INVERSE LAPLACE TRANSFORM AND TIME RESPONSE
MAGNITUDE
2.5
2.25
2.
1.75
1.5
1.25
1.
.75
.5
.25
0.
0.
.4
.8
1.2
1.6
2.
2.4
2.8
3.2
3.6
4.
4.4
4.8
5.2
5.6
6.
TIME - (SEC)
12/20/88

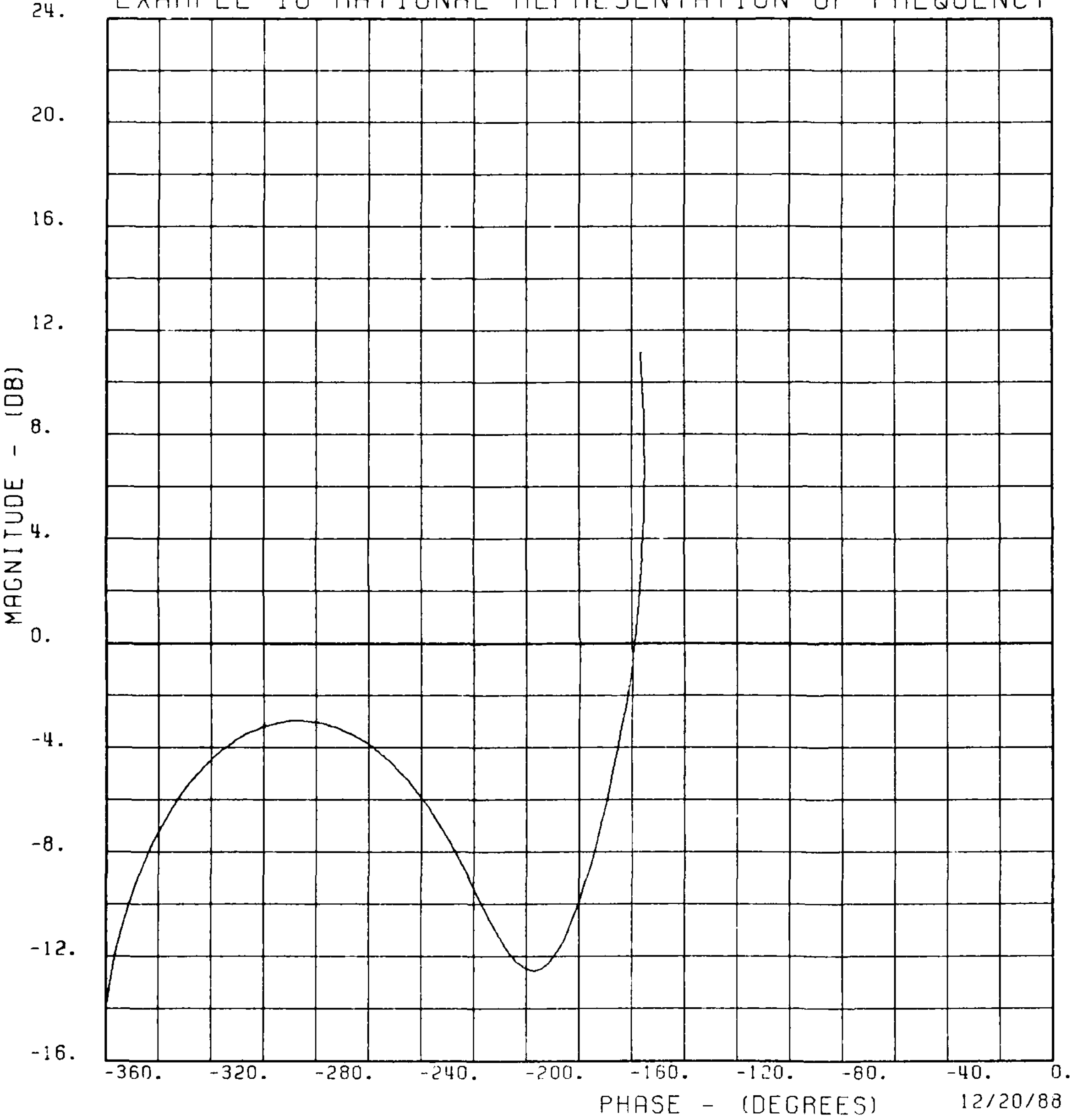
EXAMPLE 7 INVERSE Z TRANSFØRM AND TIME RESPØNSE



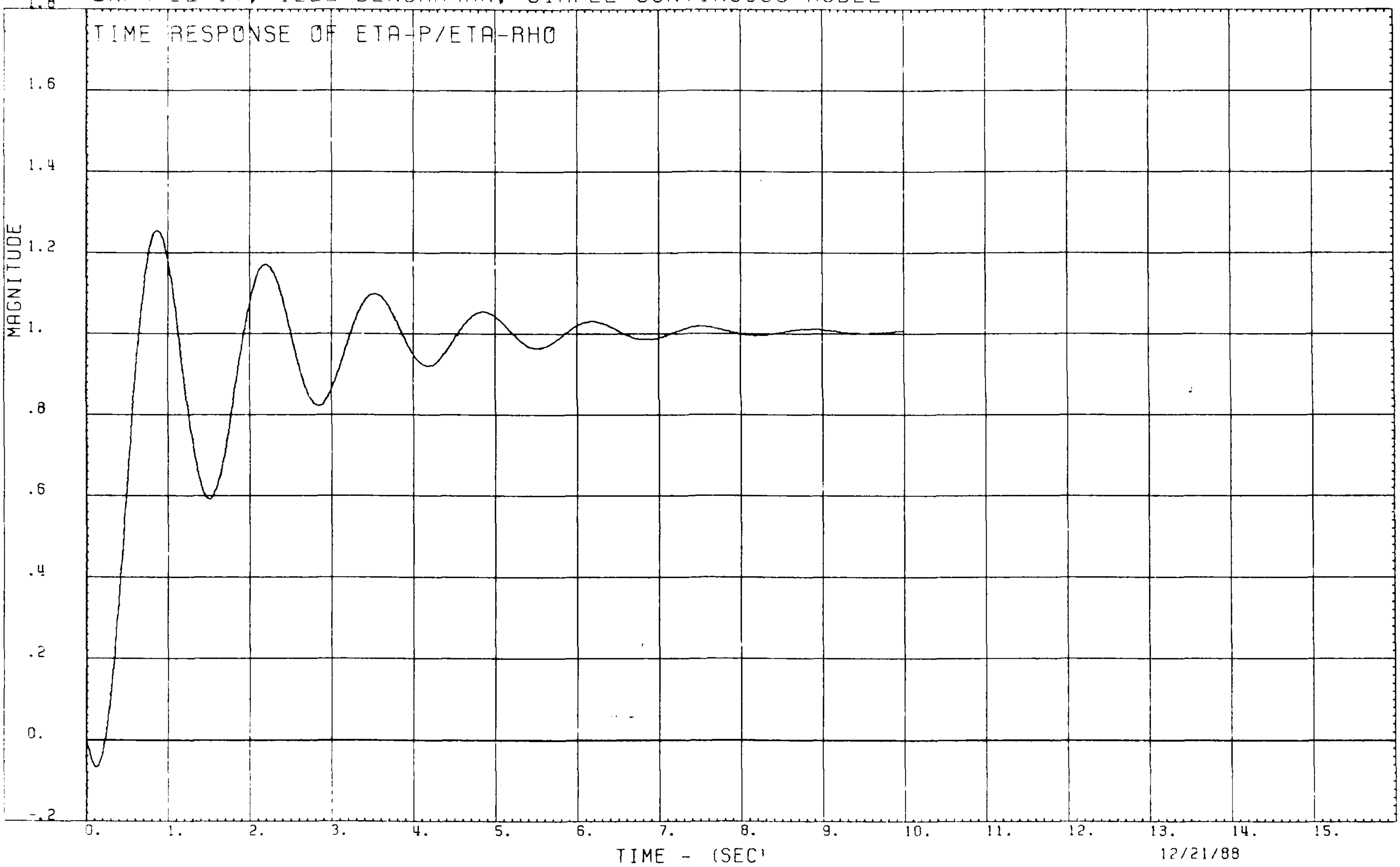
EXAMPLE 9 MULTIRATE FREQUENCY RESPONSE BY FREQUENCY DECOMPOSITION

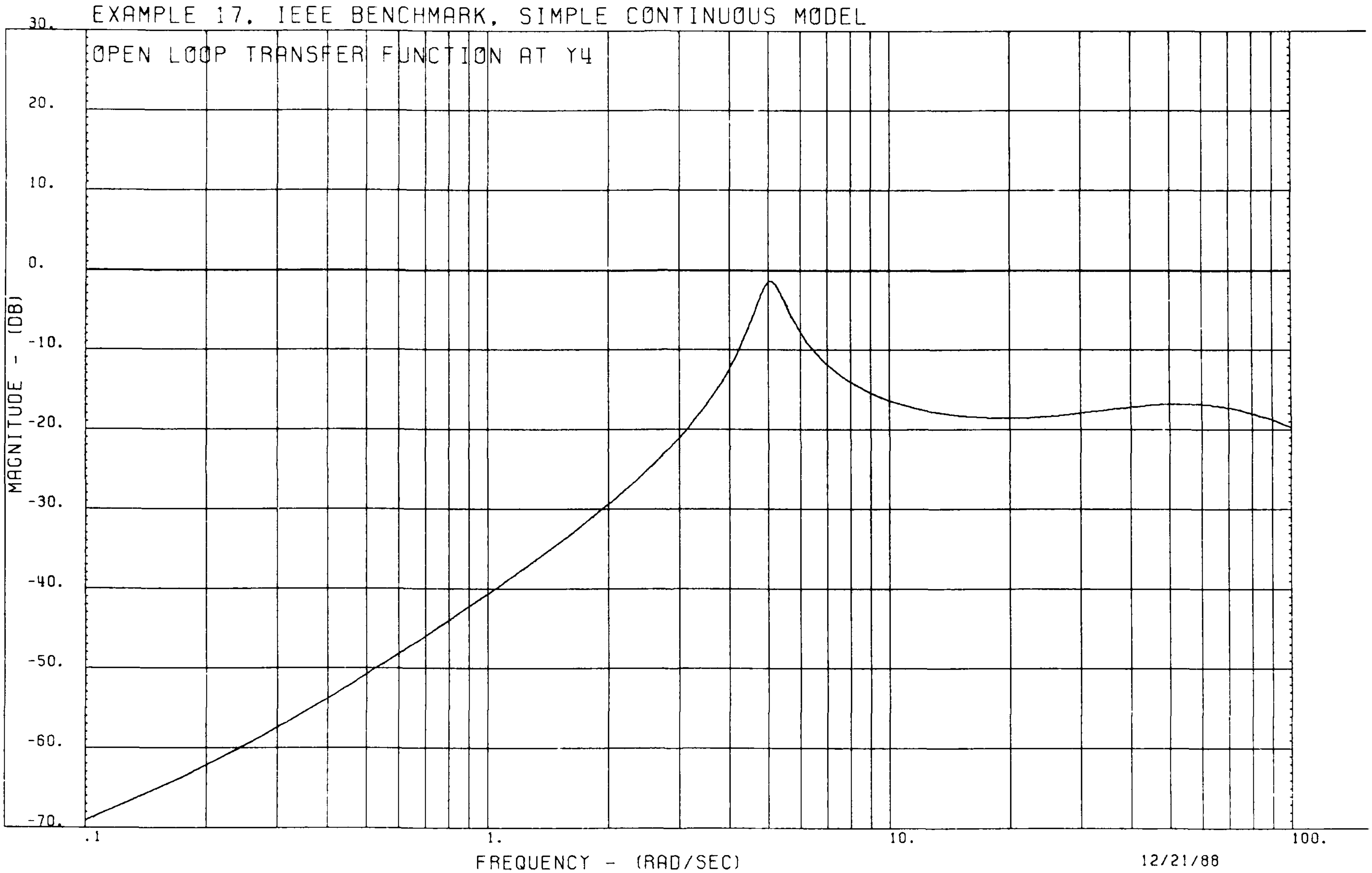


EXAMPLE 10 RATIONAL REPRESENTATION OF FREQUENCY DECOMPOSITION METHOD



EXAMPLE 17, IEEE BENCHMARK, SIMPLE CONTINUOUS MODEL

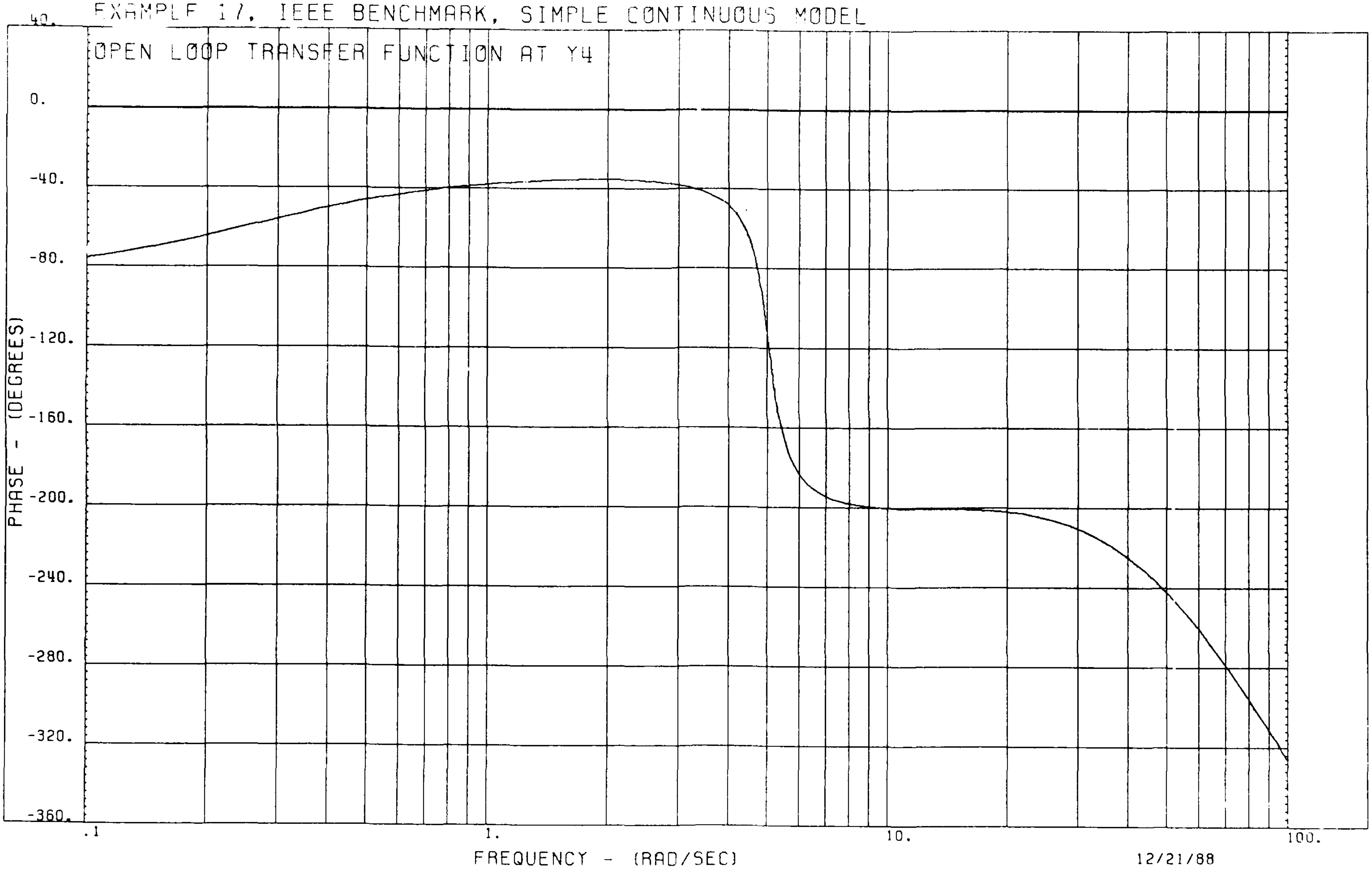


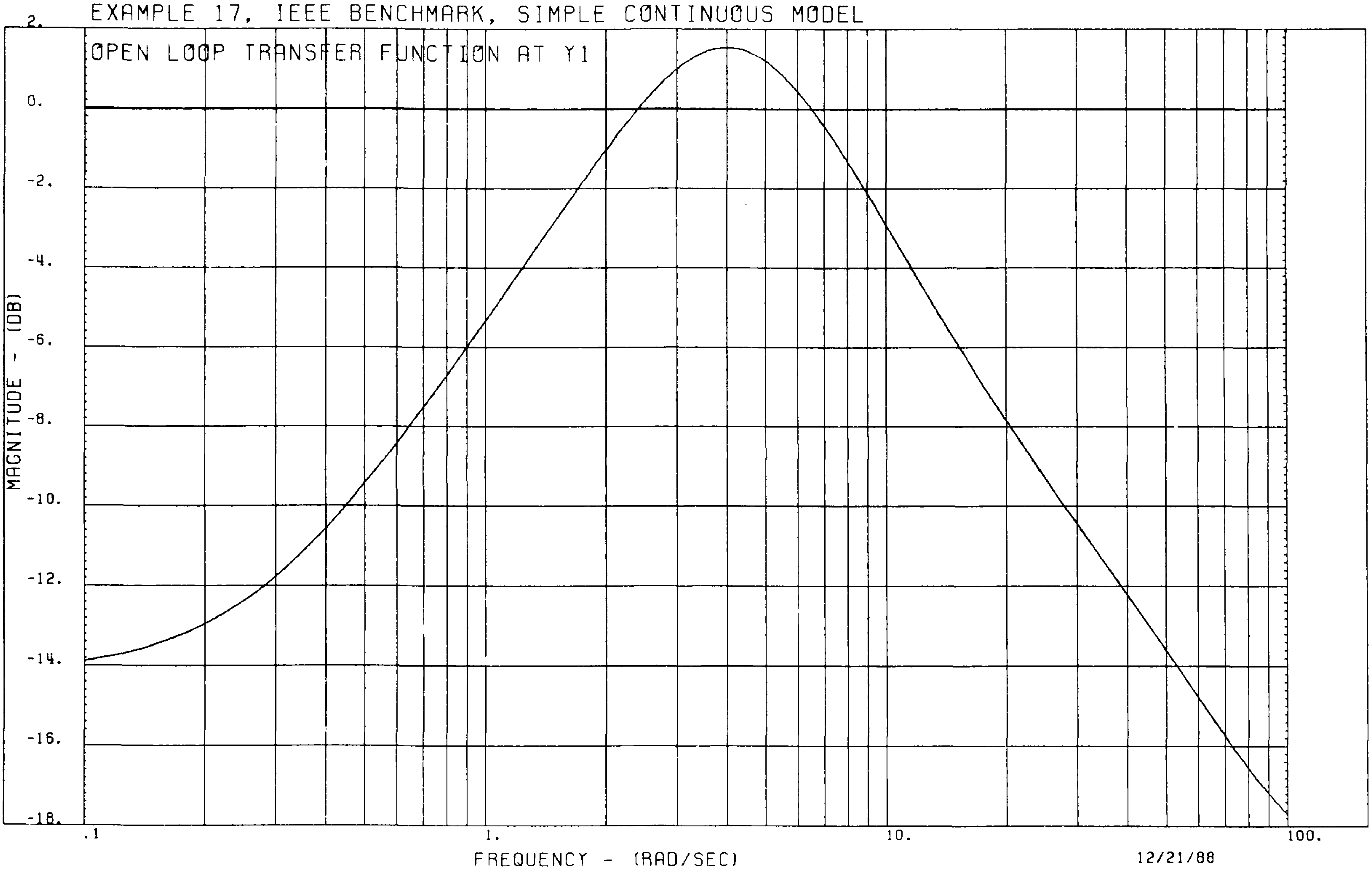


EXAMPLE 17. IEEE BENCHMARK, SIMPLE CONTINUOUS MODEL

OPEN LOOP TRANSFER FUNCTION AT Y4

12/21/88


EXAMPLE 17, IEEE BENCHMARK, SIMPLE CONTINUOUS MODEL
OPEN LOOP TRANSFER FUNCTION AT Y4
PHASE - (DEGREES)
40.
0.
-40.
-80.
-120.
-160.
-200.
-240.
-280.
-320.
-360.
.1
1.
10.
100.
FREQUENCY - (RAD/SEC)
12/21/88

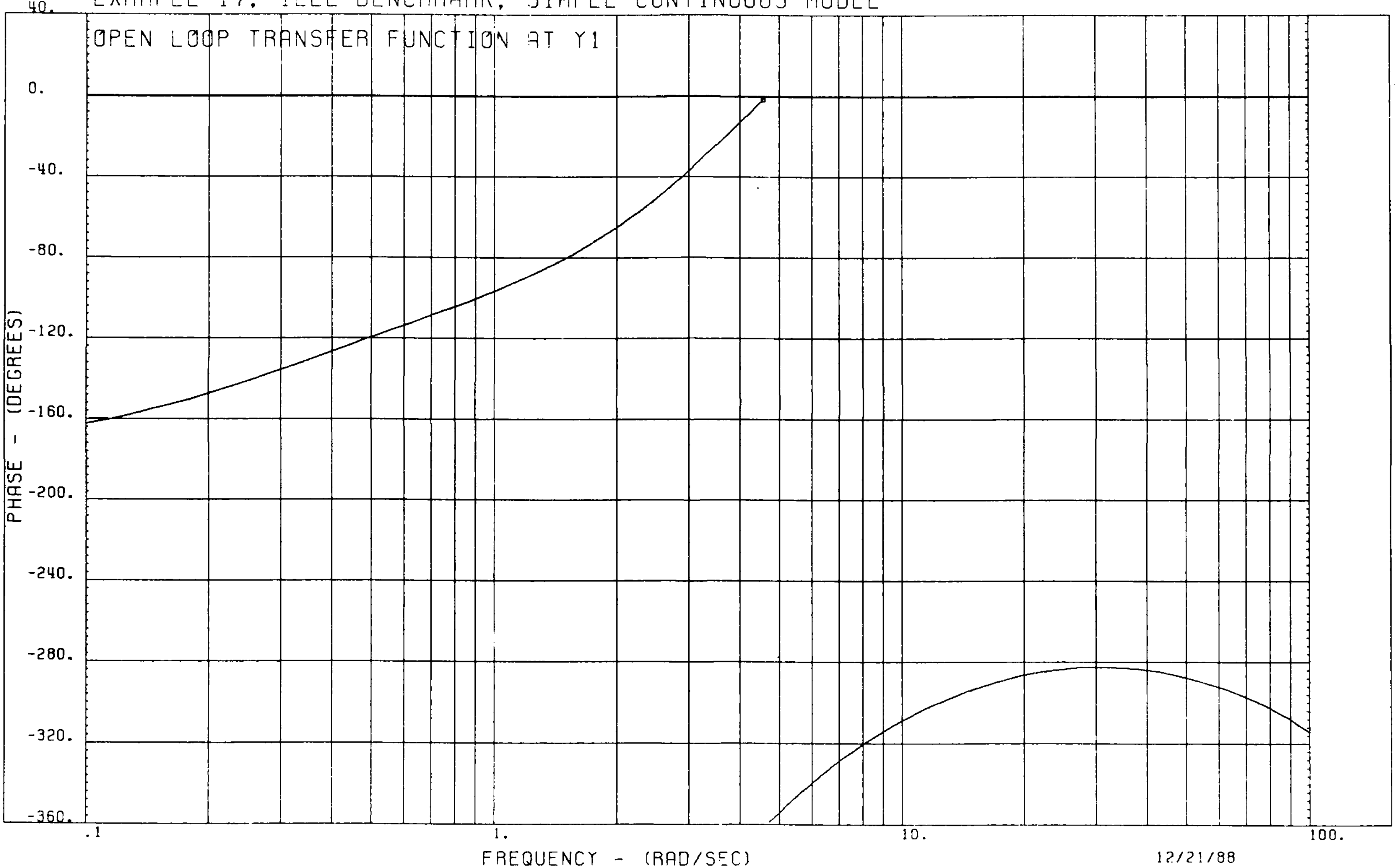


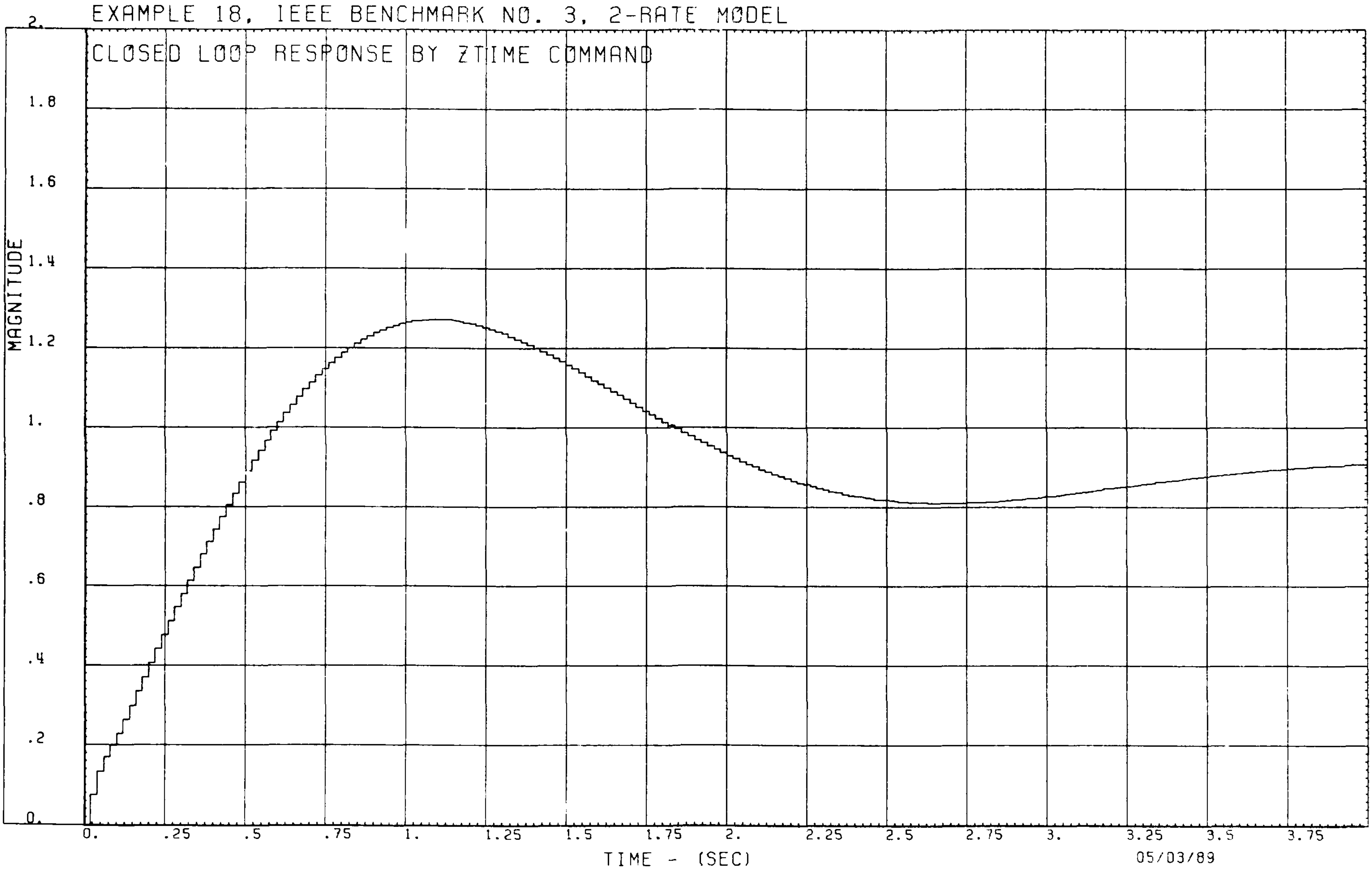
EXAMPLE 17, IEEE BENCHMARK, SIMPLE CONTINUOUS MODEL

OPEN LOOP TRANSFER FUNCTION AT Y1

EXAMPLE 17, IEEE BENCHMARK, SIMPLE CONTINUOUS MODEL

OPEN LOOP TRANSFER FUNCTION AT Y1





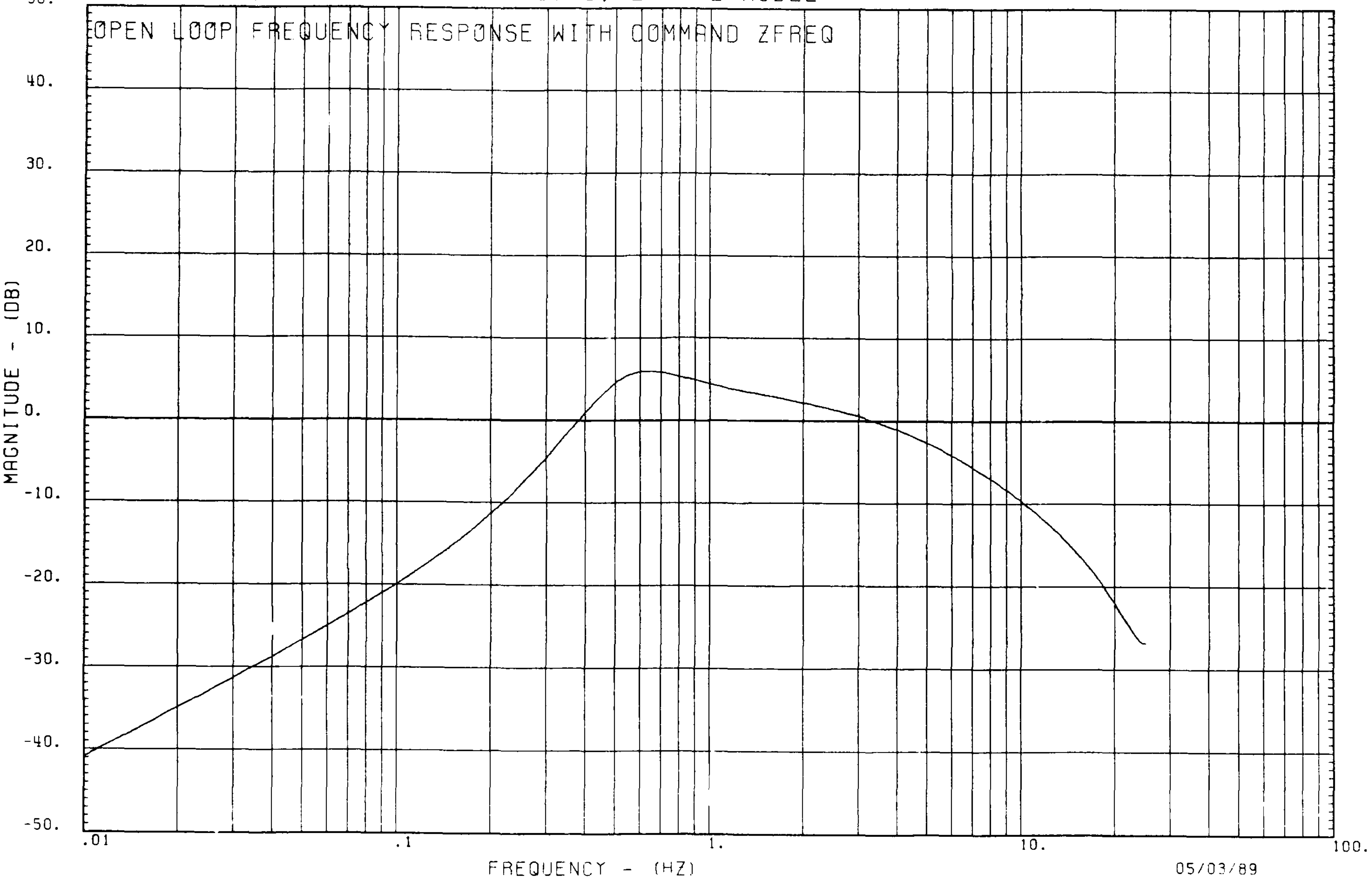
EXAMPLE 18, IEEE BENCHMARK NO. 3, 2-RATE MODEL

CLOSED LOOP RESPONSE BY ZTIME COMMAND

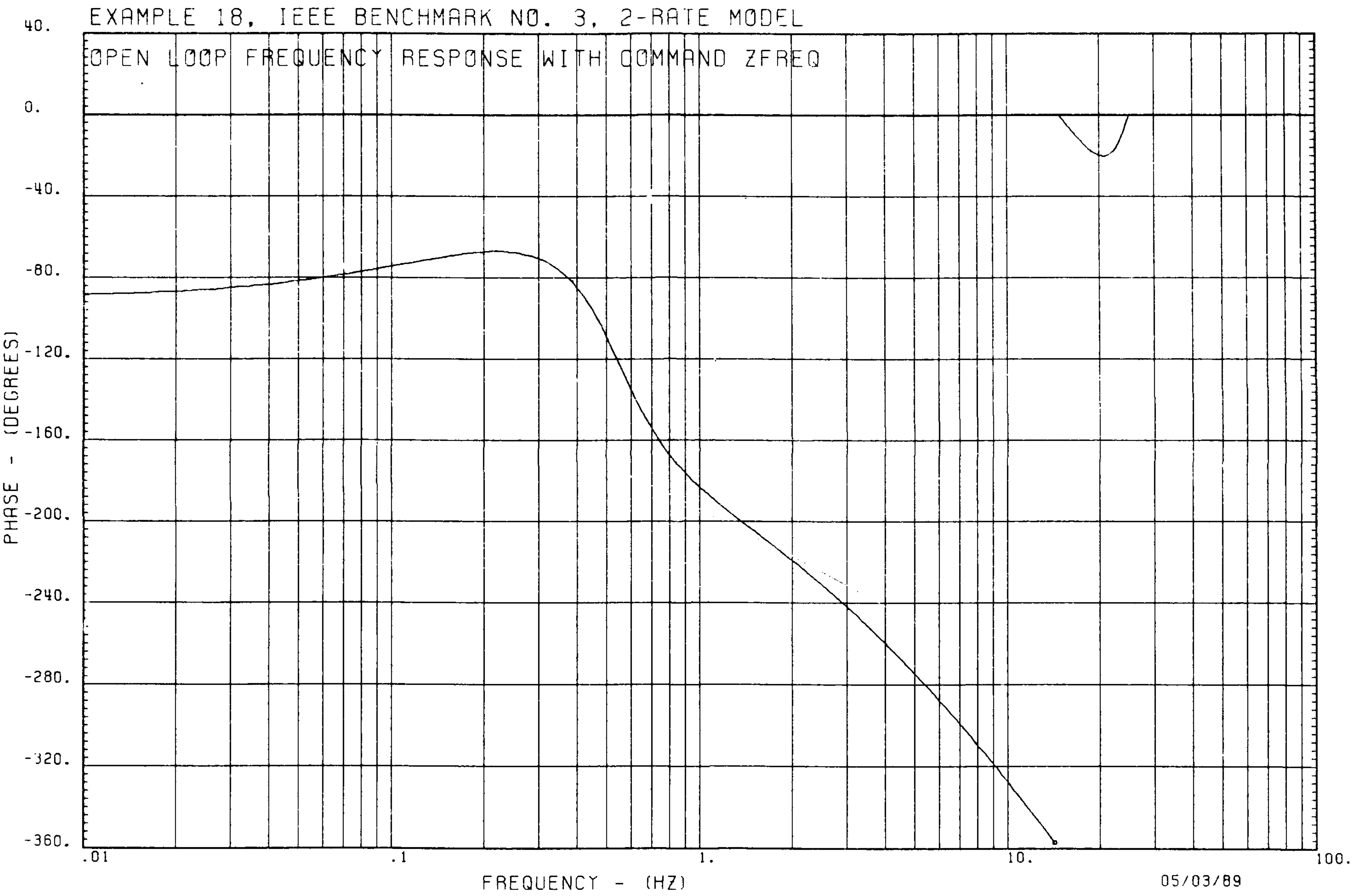
05/03/89

EXAMPLE 18, IEEE BENCHMARK NO. 3, 2-RATE MODEL

OPEN LOOP FREQUENCY RESPONSE WITH COMMAND ZFREQ



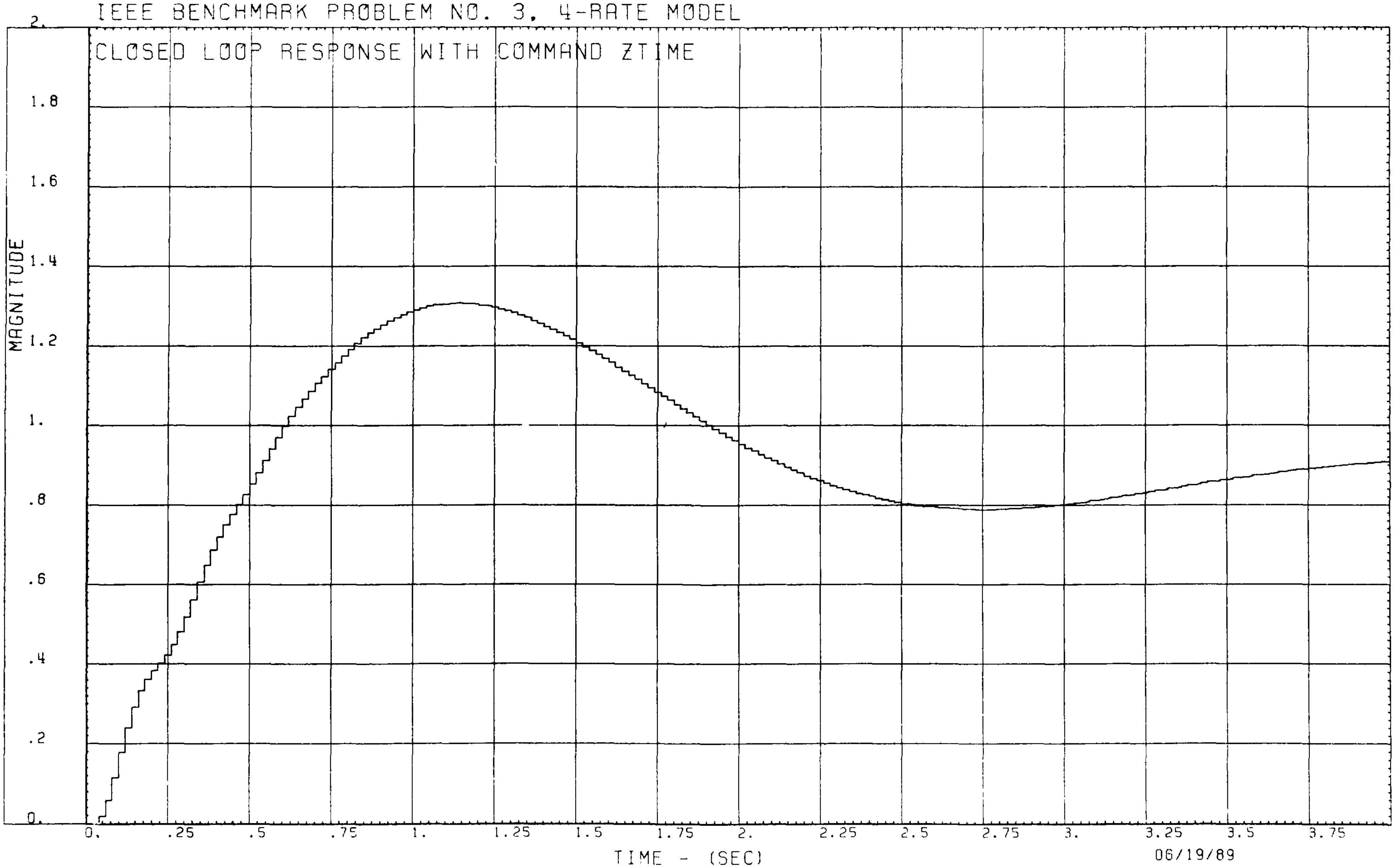
05/03/89



EXAMPLE 18, IEEE BENCHMARK NO. 3, 2-RATE MODEL

OPEN LOOP FREQUENCY RESPONSE WITH COMMAND ZFREQ

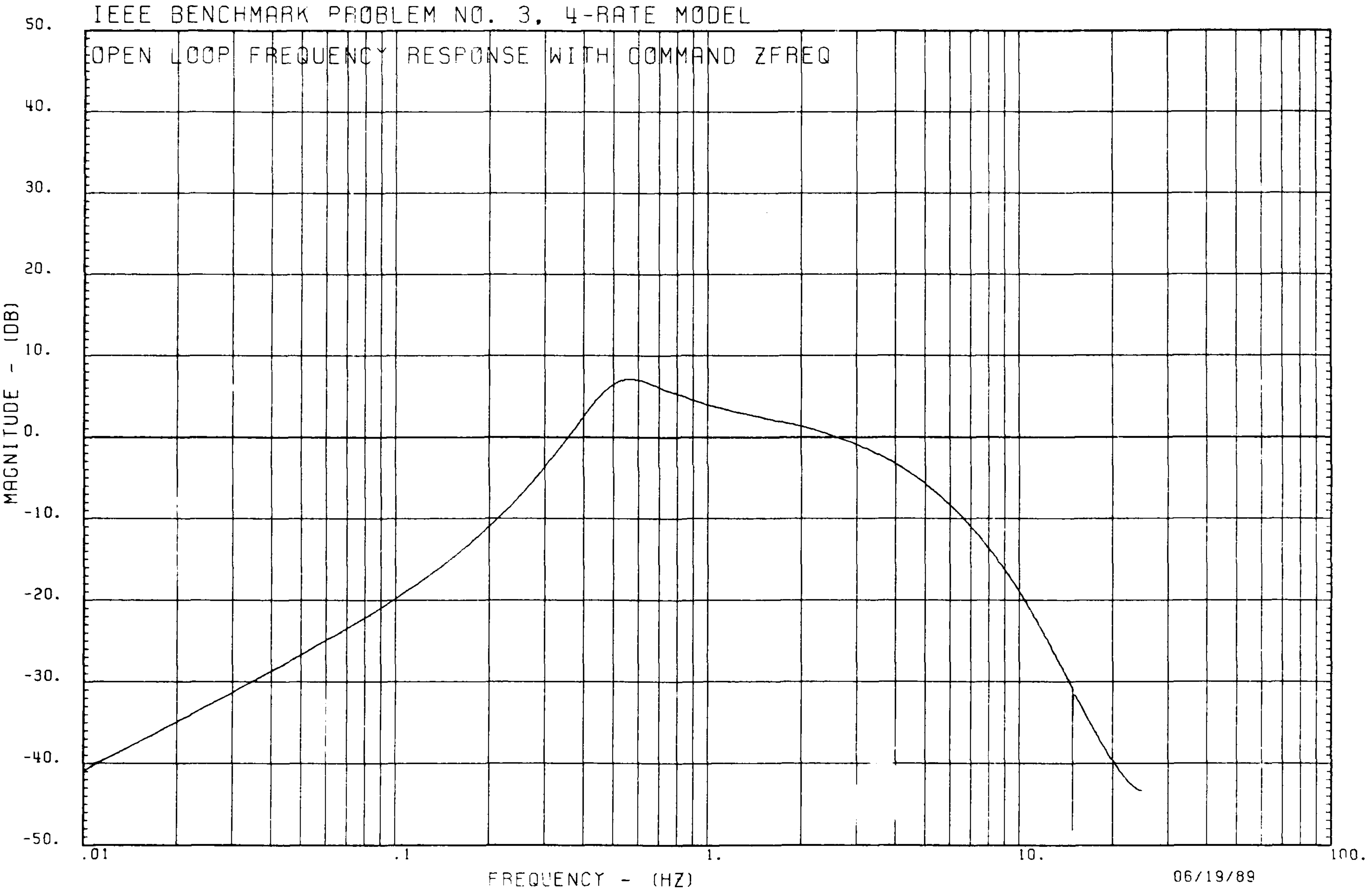
05/03/89



IEEE BENCHMARK PRØBLEM NO. 3. 4-RATE MODEL

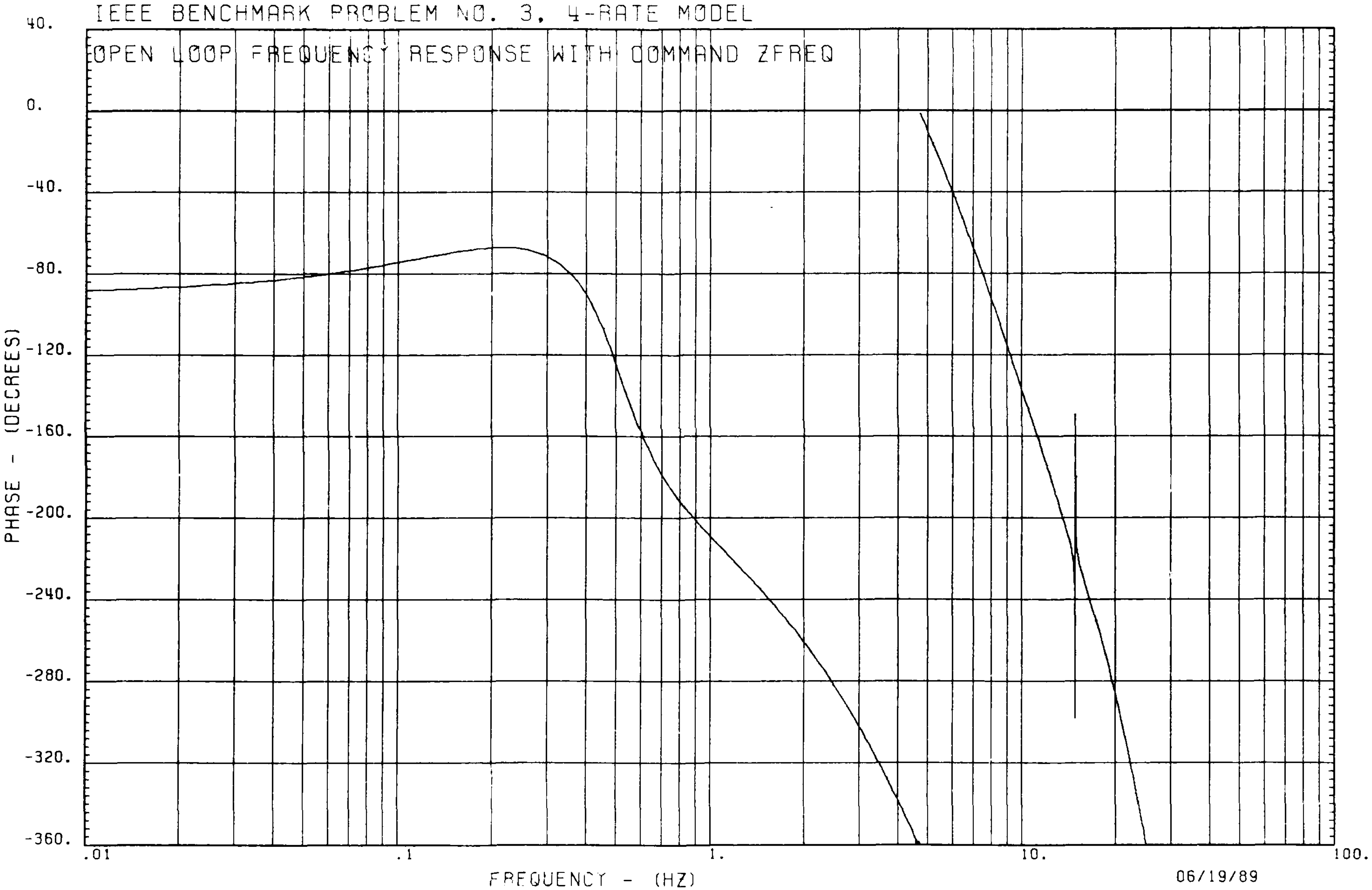
CLØSED LOOP RESPONSE WITH COMMAND ZTIME

06/19/89



IEEE BENCHMARK PROBLEM NO. 3, 4-RATE MODEL

OPEN LOOP FREQUENCY RESPONSE WITH COMMAND ZFREQ



IEEE BENCHMARK PROBLEM NO. 3. 4-RATE MODEL

OPEN LOOP FREQUENCY RESPONSE WITH COMMAND ZFREQ

Appendix I

Program Availability

The source code for this program is available to agencies supporting DOD projects and studies. The requestor, however, should be aware that some non ANSI FORTRAN 77 code are used. Installation on computers other than the CDC 176 or the CRAY XMP-14 will require some modifications. The following facts will be of interest if modifications are to be made:

1. The code was developed and modified extensively over a number of years. In many cases the code consists of a series of patches which could be cleaned up to improve both efficiency and readability. Unless significant improvements in execution time or in accuracy can be made, there are no immediate plans to do so.

2. The interactive (CDC) and batch version code share many of the same routines. Since Interactive LCAP2 runs on the CDC INTERCOM within a constraint of 230K SCM (small core memory), the segment loader was used. Since available memory in Interactive LCAP2 was already tight before the automated analysis method using transfer function connection blocks was implemented, subsequent code development had to be "shoe-horned" in to fit into the 230K SCM. This, of course, increased the complexity of the code. Non ANSI FORTRAN LEVEL 2 LCM (large core memory) code was used in some routines to enable the program to stay within the 230K SCM limit. When the batch version was converted to the CRAY these LEVEL 2 variables were changed to regular memory.

3. Evaluation of the determinant of a complex matrix is written in assembly language in the CDC version. This can be replaced with code from LINPACK.

4. Hardcopy graphics routines called by LCAP2 subroutines are in-house system routines which will not be included as part of the source code.

This program, as well as the user's guide, is in a continuous process of evolution and development. For these reasons, this program and related materials will be made available with the understanding that no warranty, expressed or implied, is made by The Aerospace Corporation as to the accuracy and functioning of the program and related materials and that no responsibility for program maintenance is implied.

A nominal handling fee for reproduction and handling will be charged. Request for a copy of this program should be addressed to:

Administrator
Information Processing Division
The Aerospace Corporation
2350 E. El Segundo Bivd.
El Segundo, California 90245

Appendix J

References

1. Lee, E. A., "Linear Controls Analysis Program (LCAP) Users Guide," The Aerospace Corporation, TOR-0077(2442-23)-1, 5 October 1976.

2. Lee, E. A., "LCAP2 - Linear Controls Analysis Program, Vol I: Batch LCAP2 User's Guide, and Vol II: Interactive LCAP2 User's Guide," The Aerospace Corporation, TR-0084(9975)-1, 15 November 1983.

3. Kalman, R. E., and J. E. Bertram, "A Unified Approach to the Theory of Sampling Systems," Journal of Franklin Institute, May 1959, pp. 405-435.

4. Hawley, P. A. and T. R. Stevens, "Two Sets of Benchmark Problems for CACSD Packages," Proceedings of the Third IEEE Control Systems Society Symposium on Computer-Aided Control System Design, Arlington, VA, September 1986.

5. Frederick, D. K., Rimer, M., and Huang, C., "IEEE CACSD Benchmark Problem No. 3, Multi-rate Sampled Data Autopilot Analysis,", 15 February 15, 1989.

6. "UPDATE Version 1 Reference Manual," 60449900, Control Data Corporation, 1984.

7. "UPDATE Reference Manual," SR-0013, CRAY Research, Inc., 1986.

8. Laub, A. J., "Efficient Multivariable Frequency Response Computations," IEEE Trans, Automatic Control, AC-26 (1981), pp 407-409.

9. Holt, J. F., "ASC MULE, General Root Finding Subroutine," Aerospace Corporation, TOR-0073(9320)-9, March 1973.

10. Frisch, H.P. and Bauer, F.H., "SAMSAN Version 2 Users's Guide, Modern Numerical Methods For Classical Sampled System Analysis," NASA/Goddard Space Flight Center, January 1984.

11. Garbow, B.S., Boyle, J.M., Dongarra, J.J., and Moler, C.B., "Matrix Eigensystem Routines - EISPACK Guide Extension," Vol. 51, Springer-Verlag, 1977.

12. Lee, E.A., "High Accuracy Computation of Z-Transform with Delay Using Bilinear Transformation," Aerospace Corporation, ATM-67(2116-60)-94, 11 April 1967. (internal distribution only)

13. Saucedo, R., and Schiring, E.E., Introduction to Continuous and Digital Control Systems, Macmillan, 1968, pp. 687-689.